Programmable Logic Development System

Operator's Manual

DATA I/O

Data I/O has made every attempt to ensure that the information in this document is accurate and complete. However, Data I/O assumes no liability for errors, or for any damages that result from use of this document or the equipment that it accompanies.

Data I/O reserves the right to make changes to this document without notice at any time.

ORDERING INFORMATION

When ordering this manual use part Number 981-0142-005.
Applies to Engineering Part Number 950-1942-008.

SAFETY SUMMARY

General safety information for operating personnel is contained in this summary. In addition, specific WARNINGS and CAUTIONS appear throughout this manual where they apply and are not included in this summary.

Definitions

WARNINGS statements identify conditions or practices that could result in personal injury or loss of life.

CAUTION statements identify conditions or practices that could result in damage to equipment or other property.

Symbols

⚠ : This symbol appears on the equipment and indicates that the user should consult the appropriate manual for further detail.

∿ : This symbol stands for Vac. For example, 120V∿ = 120 Vac)

Power Source

Check the voltage selector indicator (located inside the rear panel) to verify that the product is configured for the appropriate line voltage.

Grounding the Product

The product is grounded through the grounding conductor of the power cord. To avoid electric shock, plug the power cord into a properly wired and grounded receptacle only. Grounding this equipment is essential for its safe operation.

Power Cord

Use only the power cord specified for your equipment.

Fuse Replacement

For continued protection against the possibility of fire, replace the fuse only with a fuse of the specified voltage, current and type ratings.

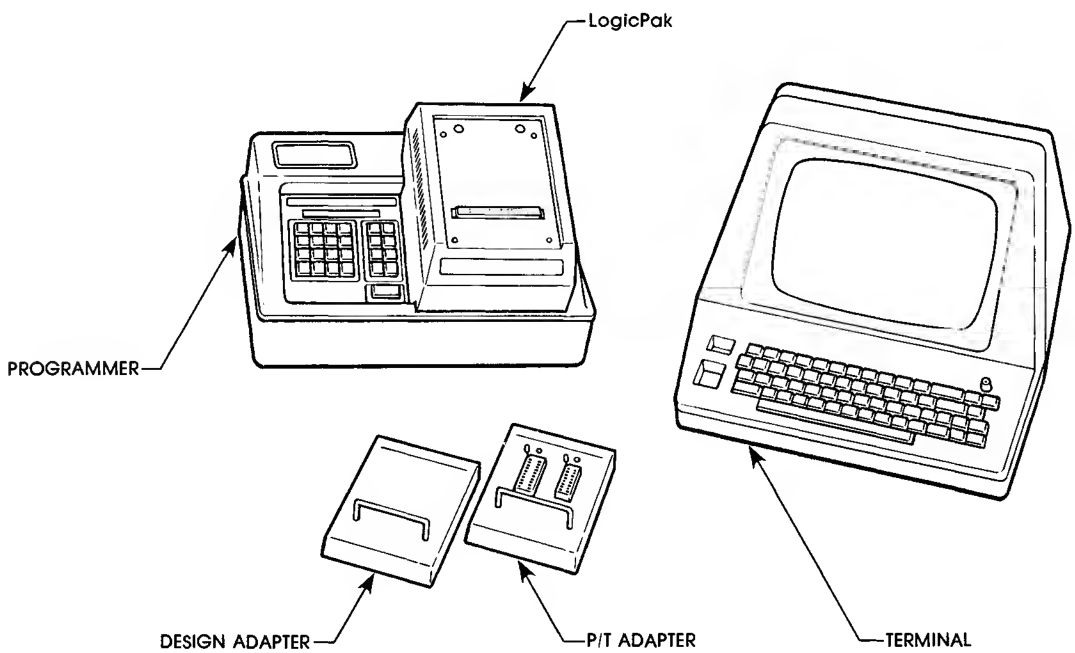
Servicing

To reduce electric shock, do not perform any servicing other than that described in this manual.

TABLE OF CONTENTS

INTRODUCTION

Data I/O's Programmable Logic Development System (PLDS) provides data development, programming, and testing support for logic device families from most semiconductor manufacturers. This modular system comprises a programmer, the LogicPak™, adapters, and a terminal as shown in the accompanying figure.



This manual contains the operational procedures of the LogicPak as well as the necessary operational information of the other components of the PLDS. Included in this manual are instructions on:

- GETTING STARTED—A sample session. Includes instructions on powering up the programmer, installing the LogicPak, installing an adapter, inserting a device, and programming the device.
- PROGRAMMING—Information about programming devices, including a list of general programming notes. Also includes information on editing, verifying data integrity, and serial port data transfer.
- REMOTE CONTROL—Provides descriptions of the remote control options available.
- SYSTEM COMMANDS—Details the select codes available from the programmer keyboard and the terminal.
- INDEX—An alphabetical guide to all the major topics covered in the manual.
- APPENDICES—A collection of reference material.

  A. Error Codes—Describes the PLDS error code displays, and corrective action.

  B. Test Modes—Provides description and operating information on the test options of the LogicPak.

  C. JEDEC Format—Provides an overview of the JEDEC format for the transfer of information between a data preparation system and a logic device programmer.

Features

The PLDS programs a wide variety of logic devices, with a minimum of equipment changes. Because P/T (program/test) adapters configure the LogicPak's hardware/firmware to specific manufacturers' device families, the only hardware you have to change when you change device families is the P/T adapter.

Data I/O offers three methods for data development. Data can be developed using ABEL™, a programmable logic design tool for PAL™ and IFL devices. ABEL (Advanced Boolean Expression Language) allows you to describe the desired operation of both PAL and IFL devices with Boolean equations, Boolean sets, logic function tables, or state diagram entry methods.

Two design adapters are available: the PALASM™ design adapter for PAL's and H & L design adapter for IFL devices.

The LogicPak can accept JEDEC file downloads (see Appendix C) from other development systems, such as ABEL.

Functional testing of logic devices is necessary to detect a functional failure in a device which could pass a routine fuse-verify test. There are two methods of functional testing available using the LogicPak.

A structured vector test may be performed on the device automatically after programming if there are structured test vectors present in the programmer data RAM. Programmed devices can be functionally tested using an optional method developed by Data I/O called the Logic Fingerprint test. This test finds defective devices using a signature analysis technique.

System Overview

The Data I/O 303A LogicPak contains all the common electronics and firmware for the programming and testing of logic devices. Any electronics or firmware unique to a specific-device family, or device within a family, are resident in P/T adapters that plug into the LogicPak. Families with more than one pin number series (e.g., PAL 20 and PAL 24) have sockets to accommodate each package size. Device families that are available in different packages, such as 20-pin DIP, 24-pin DIP, 28-pin DIP, as well as leaded and leadless chip carriers, are accommodated by the P/T adapters.

To increase flexibility in waveform generation, digital- to- analog converters (DAC) control all major power supplies, with several rise and fall times selected by software.

System Capabilities and Operational Summary

The LogicPak can be used with the Model 29 Universal Programmer, the System 19, or 100A programmers (see the subsection on programmer compatibility).

To program a logic device, you must first load the desired state of the device fuses into the programmer RAM. Fuse information can be input in several forms: JEDEC file information developed with ABEL, decimal fuse number and state, H & L programming tables for IFL devices, and Boolean equations for PAL devices (PALASM). The detailed operating procedures for data development using ABEL or the design adapters are provided in the ABEL, PALASM and H & L manuals.

Once the fuse pattern is loaded into RAM, you can program the device using a P/T adapter. Thus, the design adapter can be removed from the LogicPak, and an appropriate P/T adapter can be installed allowing the fuse pattern to be programmed directly into the device and automatically tested. The programming steps are virtually the same as those taken when programming PROMs. To start the programming sequence, first enter a device code, then programming can begin (see the Getting Started section for a sample programming procedure and your programmer manual for further operating details).

Because a programmable logic device is not completely manufactured until it is programmed, the programmer must be able to perform some type of functional testing. This is accomplished by using structured vector testing and/or Data I/O Logic Fingerprint testing.

Specifying a Fusemap

The design adapter firmware translates your design into a fuse pattern and, optionally, test vectors. This fuse pattern and test data are resident in the programmer's RAM. If a fuse pattern is generated on a host system, it must use fuse numbers specified according to logic diagrams in each P/T adapter User Note and transmitted to the programmer in the JEDEC (Joint Electron Device Engineering Council) format (see Appendix C). Data I/O uses the JEDEC Logic Device Translation Format (number JC-42, 162) for serial data input and output with the LogicPak. The only exception to this is when you are using a Signetics H & L design adapter, in which case data transfer can also occur in the Signetics H & L logic format.

An alternate method of specifying the fusemap is to manually enter the fuse number and state (see the logic diagrams for each adapter in the user notes for the particular adapter) for every fuse in the device. These diagrams are the same as those in the device manufacturers' data books, but the fuse numbers have been added. Although the task is tedious, fuse numbers and states can be entered manually into the programmer's data RAM from the programmer's keyboard or from a terminal using a fuse editor. This method usually will be used only for editing fuse data because it is a long process with room for error.

The P/T Adapter

With a P/T adapter, fuse data can also be entered into the programmer's RAM by loading from a master device. Blank devices can then be programmed using the same P/T adapter, or other manufacturers' functionally equivalent second-source devices can be programmed by installing the appropriate P/T adapter. Remember that a device that has its security fuse programmed cannot be used as a master because its fuses cannot be read.

A different programmer RAM fusemap was used in some previous Data I/O logic device programming modules (950-0800, 919-1427, and 919-1542). If you have paper tapes or files that you prepared to use with these modules, you must prepare new files for use with the LogicPak unless you used the Signetics H&L logic format. The preferred translator format is the JEDEC format, but a Signetics translator is available. (The 950-0104 and 919-0045 modules use a memory map that is compatible with the LogicPak.) The 919-1427 pak used a fusemap generated to simulate a 512 x 4 PROM. Serial data were entered using a standard PROM data translation format. Again, tapes must be regenerated for use with the LogicPak because of the different correspondence between fuses and bits in RAM. The fusemaps have been changed to allow for the programming of functional second-source devices from the same fusemap loaded in RAM and to avoid gaps (previously called phantom fuses) in the fusemaps.

Sumcheck

After fuse data have been loaded into the programmer from the serial port or the master device, the programmer calculates the sumcheck of the fuse data (see the next figure) and displays it (when in terminal mode, it will be displayed on the terminal). The sumcheck, which is used to verify the integrity of data transfers, is a summation of eight-bit bytes of fuse data expressed as a four-digit hexadecimal number as shown in the table.

Hexadecimal Data	Binary Data
84	10000100
C1	11000001
62	01100010
24	00100100
01CB	0000 0001 1100 1011
16-Bit sumcheck in hexadecimal notation	16-bit binary hexadecimal sumcheck

If data was loaded through the serial port and a sumcheck was sent with it, the programmer will compare the sumcheck with its own calculation. If they agree, the correct sumcheck is displayed. If they do not agree, the programmer will signal an error. Data from the serial port will also be checked for correct parity if the programmer parity switch is on (see your programmer manual).

Data from a master device are loaded into RAM by installing the correct P/T adapter on the LogicPak and performing a load operation. Any information in the source buffer (Boolean equations, function tables, or test vectors) will not be affected. The source buffer is used in conjunction with the design adapters as a holding area for the translation of the Boolean equations into the device fuse map.

Test Capabilities

With the LogicPak installed on the programmer, when power is applied, the programmer will perform a series of self-tests to ensure functionality. A failure will display an error code (see Appendix A). To isolate the problem to its source, refer to the maintenance manual.

Functional Testing verifies that a programmed device will perform as intended. Descriptions of how to test devices is in the System Commands section. Additional information is given in Appendix B.

- ☐ Fuse Verity—A test to verify that the fuse pattern in the device and the programmer RAM are the same. This test is run automatically during the programming cycle.
- ☐ Structured Test—When test vectors are entered, the structured tests are always performed prior to the Logic Fingerprint test. If no test vectors are present, only a fuse verify and a Logic Fingerprint test (if enabled) will be performed.
- ☐ Logic Fingerprint Test—This is the easiest test to use and is flexible enough to be applicable to a wide variety of logic devices. Refer to your P/T adapter User Note for Logic Fingerprint Test limitations.

Programmer Compatibility

To be compatible with the LogicPak, your programmer may require a hardware and/or firmware update, depending on the model, configuration, and age. The information that follows will help you determine whether your programmer requires updating. If you find that your programmer does require updating, contact your nearest Data I/O customer support center. (A list of customer support centers is at the back of this manual.)

The 29B programmer has no special compatibility requirements for proper operation with the LogicPak. Refer to your P/T adapter User Notes for any additional compatibility requirements that may be necessary.

- ☐ System 17—The System 17 must be converted into a System 19 with the latest firmware installed and latest hardware modifications.
- ☐ System 19—Check to determine whether your System 19 contains a 702-1520 or 702-1980 controller board by performing the following steps:

  1. Remove the programming pak.
  2. Remove the metal or plastic shield (if any).
  3. Count the number of EPROM firmware sockets located just behind the pak interface connector. If there are four sockets, it is a 702-1520 board. If there are eight sockets, it is a 702-1980 board.

  If your System 19 contains a 702-1520 controller board, check the modification status sticker on the bottom of the programmer. If the sticker is not there, or if the "1" is marked off, your System 19 requires hardware and firmware updating; contact the nearest Data I/O customer support center. If "2" is marked, your System 19 is compatible with the LogicPak. If your System 19 contains a 702-1980 controller board, it may require a firmware update. To display the configuration number of the firmware in your programmer, key in "SELECT-B2-START" and observe the display.

  If the configuration number displayed is "3599" or "CC8B," your firmware needs updating.

- ☐ Model 29A Universal Programmer—To be compatible with the LogicPak, the Model 29A programmers must have Rev (revision) C or later firmware. To determine the configuration of the firmware in your Model 29A, key in "SELECT-B2-START" and observe the display. If the hexadecimal number matches one listed in the following table, your firmware needs to be updated.

Model 29A and 100A Programmers Requiring a Firmware Update

Model	Rev	Configuration Number
29A	A	1ECA
	B	20A4
29A (with computer remote control)	A	BB41
	B	C00B
100A	A	917F
	B	9405
	C	9DEE
	D	9BED

- ☐ 100A Production Programmer—To be compatible with the LogicPak, the 100A programmers must have Rev E or later firmware. To determine the configuration of the firmware in your 100A, key in "SELECT-10-START" and observe the display. If the hexadecimal number display matches one listed in the table, your firmware needs to be updated.

Specifications

The physical and environmental specifications of the LogicPak are:

- altitude (operating): sea level to 3 km (10,000 ft)
- humidity (operating): 90% maximum (noncondensing)
- humidity (storage): 95% maximum (noncondensing)
- temperature (operating): 5 to 45 °C (41 to 113 °F)
- temperature (storage): -40 to 70 °C (-40 to 158 °F)
- weight: 1.6 kg (3 lb, 8 oz)
- dimensions: 17.9 x 17.3 x 21.7 cm (7.05 x 6.81 x 8.54 in.)

Applications

As Data I/O increases the capabilities of the LogicPak to program new or additional devices, firmware updates will be available for existing adapters to add new devices to existing-device families. New adapters may also be added to the LogicPak to accommodate new device families.

Warranty and Customer Support

Data I/O equipment is warranted against defects in materials and workmanship. The warranty period of one year, unless specified otherwise, begins when you receive the equipment. Refer to the warranty and inside the back cover of this manual for information on the length and conditions of the warranty. For warranty service, contact your nearest Data I/O Customer Support Center.

Data I/O maintains customer support centers throughout the world, each staffed with factory-trained technicians to provide prompt, quality service. This includes not only repairs, but also calibration of all Data I/O products. A list of all Data I/O customer support centers is located in the back of this manual.

Ordering

Orders made with Data I/O must contain the following information:

- ☐ Description of the equipment
- ☐ Quantity of each item ordered
- ☐ Shipping and billing address of firm, including ZIP code
- ☐ Name of person ordering equipment
- ☐ Purchase order number
- ☐ Desired method of shipment

Options

The following items can be ordered by contacting your Data I/O service representative.

Options	
303A-001	IFL P/T Adapter
303A-002	MMI/National PAL P/T Adapter
303A-003	Harris P/T Adapter
303A-004	AMD PAL P/T Adapter
303A-006	Texas Instruments P/T Adapter
303A-007	Harris CMOS P/T Adapter
303A-008A	32R16 P/T Adapter (DIP + LCC)
303A-008B	32R16 P/T Adapter (DIP + NLCC)
303A-009	CMOS P/T Adapter
303A-010	Altera/Intel 40-Pin P/T Adapter
303A-100	PALASM Design Adapter
303A-101	Signetics H & L Design Adapter

GETTING STARTED

This section explains how to get started using your LogicPak and a Model 29 programmer. Refer to the programmer manual for the procedure to perform a load operation. This section also provides the necessary procedures for connecting the other components of the PLDS. Included here are complete procedures for powering up and for programming a device from a master device and using your programmer keyboard. For details on operating your programmer, refer to your programmer manual and the sections on Programming and Remote Control of this manual.

This section includes the following information:

- Installing the LogicPak on the Programmer
- Installing a P/T Adapter on the LogicPak
- Sample Programming Session

Power Connection

Since the LogicPak is a plug-in module that is inserted on your programmer, the power connection procedures you will follow are those for your programmer. The LogicPak alone has no power connections. Refer to your programmer manual for the procedures.

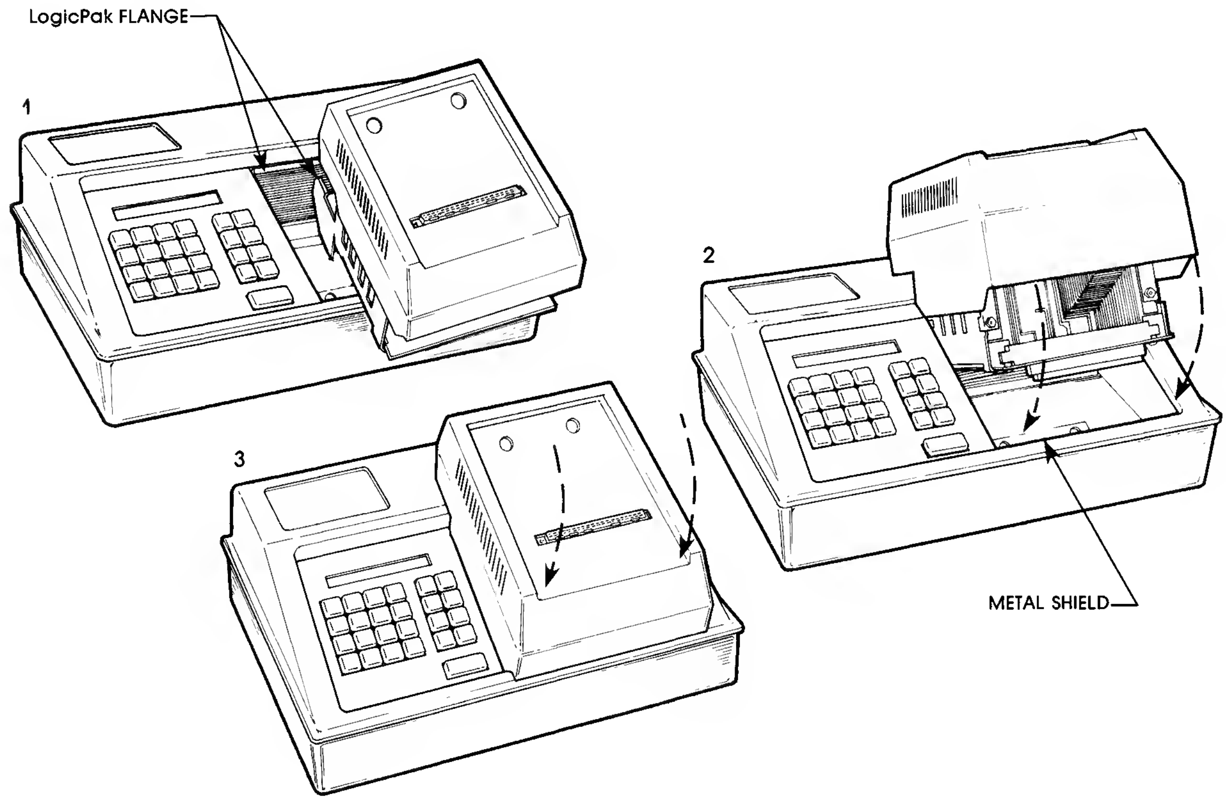
LogicPak Installation

The LogicPak may be installed and removed with the programmer's power on; this feature allows you to retain data in RAM during module changes. If the programmer power is turned on before the LogicPak is installed, you will hear a beep until the LogicPak with an adapter is installed.

CAUTION

Voltage transients can cause device damage. If a P/T adapter is installed in the LogicPak, be sure that all sockets are empty when switching power on or off, before installing or removing the LogicPak or adapter.

To install the LogicPak into a programmer, refer to the figure and follow this installation procedure.



NOTE
Although the Model 29 is shown here, the insertion procedure is the same for all systems.

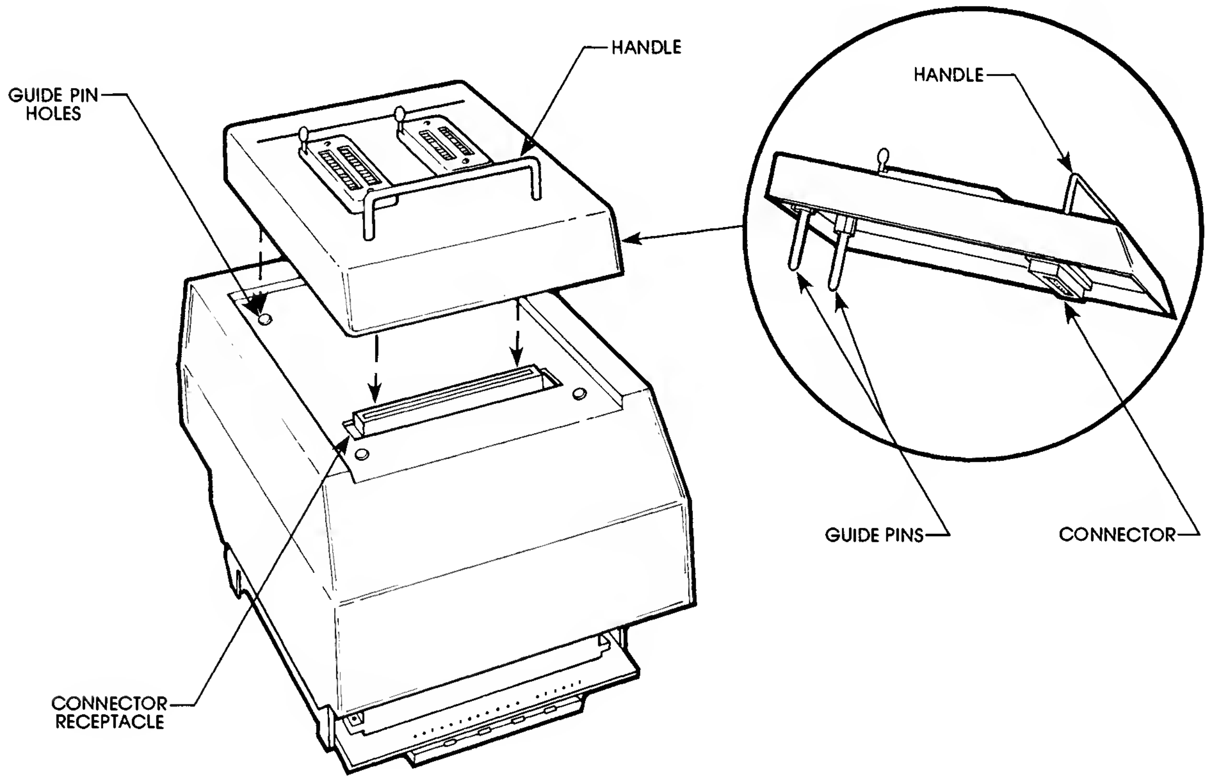
1. Slide the LogicPak into the opening in the programmer.
2. Tilt the LogicPak up, and gently push it back to hook its flange over the back edge of the programmer opening.
3. Lower the LogicPak into position as shown in the figure.
4. Press down gently on the front of the LogicPak to ensure a good connection.

CAUTION

Be careful when inserting the LogicPak. Always hold the LogicPak by the chassis and not by the adapter handle when carrying it.

Adapter Installation

To insert all adapters into the LogicPak, refer to the figure and follow this installation procedure.

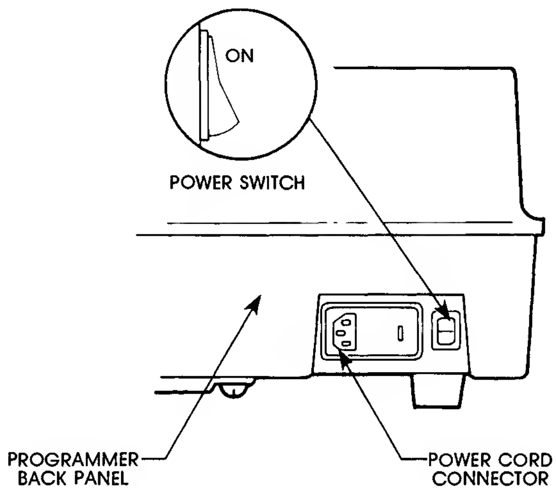


1. Check to make sure a device is not in a socket. If a device is in a socket, remove it by lifting the lever on the side of the socket and then lift the device out of the socket.
2. Align the guide pins on the underside of the adapter with the guide pin holes on the LogicPak as shown in the figure.
3. Gently set the adapter on the LogicPak.
4. Firmly press down on the front edge of the adapter to lock the connector pins into the connector receptacle.

Powering Up

The first step in getting started is powering up your programmer. Use the following procedure.

1. Check to make sure the adapter sockets are empty. If a device is in a socket, remove it.
2. Check to be sure the voltage selector is in the proper position. Plug the AC power cord into the rear of the programmer, and into a power receptacle.
3. Press the power switch at the back of the programmer to the "ON" position as shown in the figure below.



When the programmer is powered up, it automatically performs a self-test routine. This will take a few moments and the programmer will display indications that the test is being performed.

Sample Programming Session

The following steps describe how to program a 16R8 device using a master device (a part that has been previously programmed and is used as a "master" to program other parts). This procedure assumes that the LogicPak is installed in the programmer. The programming section in your programmer manual gives complete descriptions of the commands and procedure.

NOTE

This programming session assumes operation and key entry from the programmer front panel. These programming operations can also be executed from the terminal and are discussed in the System Commands section of this manual.

1. Press [COPY] [DEVICE] [RAM] [START]

   to prepare the programmer to transfer the fuse pattern data from the master device to the programmer's RAM. Refer to your P/T adapter manual or User Note for complete listings of family/pinout codes for each adapter. Find the code numbers corresponding to the device number for the manufacturer of the device. These codes are found under the column "family/pinout". The programmer will display

   FAM 00 PIN 00

2. Press [2] [2] [2] [4]

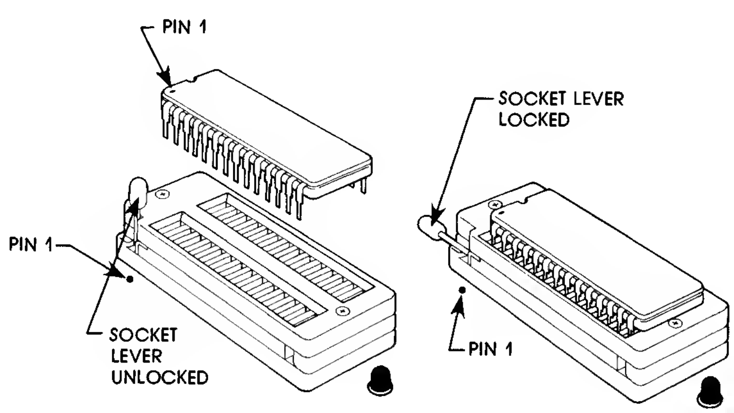
   the family/pinout code for the MMI 16R8 device. The programmer will then display

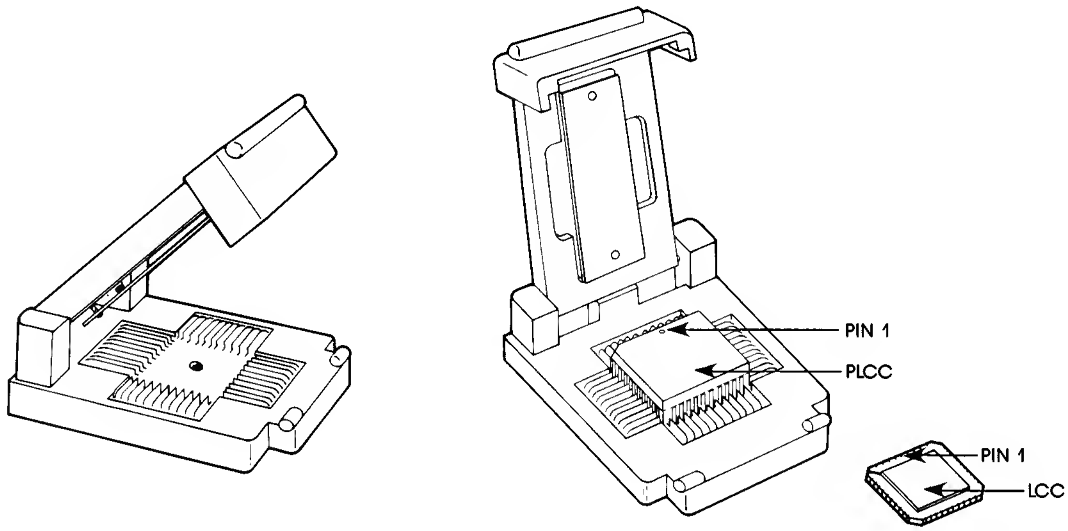
   FAM 22 PIN 24

NOTE

The 303A-008 P/T adapter only programs the 32R16 PAL; therefore the keyboard or terminal will always display "2247", the family and pinout code for the 32R16 PAL. These codes are set automatically when power is turned on. No other codes will be accepted.

3. Lift up the lever on the socket that has an illuminated LED below it (see figure). Line up pin 1 of the device so that it is nearest the pin 1 indication dot and set the device into the socket. Press down on the lever to lock the device in place.





NOTE
For LCC devices, orient pin 1 according to the drawing in your P/T adapter User Note.

4. Press [START]. The programmer will display

LOADING DEVICE ◻
LOAD DONE XXXX

NOTE
XXXX is the sumcheck of the device. See step 8 for more information.

5. Lift up the socket lever and remove the master device from the socket. The master device fuse data is now transferred to RAM. The next part of the procedure transfers that fuse data to the blank device.

6. Press [COPY] [RAM] [DEVICE] [START]

   to prepare the programmer to transfer the fuse data from RAM to the blank device. The programmer will display

FAM 22 PIN 24

7. Line up pin 1 of the blank device so that it is nearest the pin 1 indication dot and set the device into the socket. Press down on the lever to lock the device in place.

8. Press [START]. The programmer will display

TEST DEVICE ◻
PROGRAM DEVICE ◻
VERIFY DEVICE ◻
PRG DONE 01 XXXX

NOTE
XXXX in the above display represents the device's sumcheck, the hexadecimal sum of all the bytes in the device. The number displayed should match the sumcheck displayed during step 4 of this procedure. "PROG DONE 01" means that 1 device has been programmed.

9. Lift up the socket lever and remove the device from the socket. The device is now programmed.
10. To program another device, simply place it in the socket and press START.

PROGRAMMING

After fuse data development, the next step in the programming sequence is programming the logic devices. The LogicPak applies the manufacturer's specific algorithms to blow fuses in the logic device according to the fuse pattern data in the programmer RAM. Once you key in the operation on the programmer, programming is automatic and starts with a series of tests: backward device test, illegal-bit test, and blank check. During the backward device test, the programmer automatically checks the device's orientation in the P/T adapter socket and displays an error if it is inserted backwards. The Model 29 displays:

DEV BACKWARDS 32

The illegal-bit test checks for previously programmed bits in a nonblank device that should not be programmed according to the fuse pattern in RAM. If illegal bits exist, the Model 29 displays:

ILLEGAL BIT 21

During the blank check, the programmer searches the device for blown fuses. If any are found (and the bits are legal), the programmer will signal the operator and the Model 29 displays:

NONBLANK 20

Nonblank parts can be over-programmed by again pressing

START

If the device passes these tests, data are transferred from the programmer RAM to the LogicPak. The LogicPak then applies the programming pulses to the appropriate pins and tests the state of the selected fuse condition. If the fuse fails to program, the Model 29 displays:

PROGRAM FAIL 22

Otherwise, programming proceeds to the next fuse until all have been programmed. (With some P/T adapters, programming algorithms vary and may not display the "program fail" error message. These adapters will, however, display a "verify fail" message if a fuse fails to program.)

The Model 19 will display the following error codes:

ERR 32
ERR 22
ERR 21

A blinking display of the current RAM address and data implies the device is nonblank

Refer to Appendix A for a listing of the error codes generated by the LogicPak. If an error code does not appear there, check the programmer manual.

Editing Features

After data have been developed or loaded into programmer RAM, the PLDS provides editing features that allow you to modify the fuse pattern. These are accessed through one and two-digit commands entered on the programmer keyboard or on a terminal. Refer to the Edit Fuse Pattern subsection in the System Commands section for more details.

REMOTE CONTROL

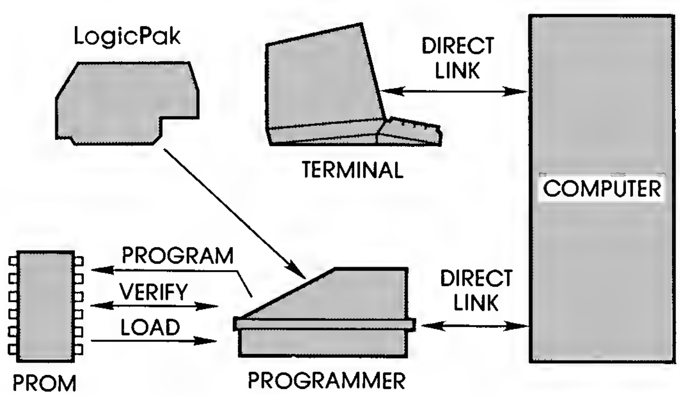
This section of the manual provides descriptions of the remote control options available for programmer/LogicPak operation.

The LogicPak uses the RS-232C serial interface of the programmer for interaction with a terminal and for communication with a host computer. For complete interconnection methods see your programmer manual.

Computer Remote Control

Computer Remote Control (CRC) is designed to enable you to control the programmer by a computer. Linked directly to the programmer, the computer generates and sends commands to the programmer, determines variables for setting programming parameters (where needed), and reacts to information, returned to it from the programmer. While these commands may be sent by an operator at a terminal, the commands and syntax were designed to be easily incorporated into a computer program.

The figure below illustrates the basic components of a logic development system under computer remote control. For interactive programs, the computer can send messages to be displayed on the programmer, and can read keystrokes from the programmer's keyboard (29B V03 or later only). The user must provide application software which will allow the computer to issue CRC commands and to interpret CRC responses. The list of commands and responses is contained in the programmer manual.



For device programming convenience, you can also control your Data I/O programmer from an IBM PC with Data I/O's menu-driven remote control software package PROMlink. Data files can be stored and retrieved from the PC, eliminating the need for master devices or paper tapes. Complete control of the programmer is also accomplished by this method.

Terminal Remote Control

System Command E1 transfers control of the programmer to the terminal. After control is transferred, the programmer will display only its action symbol. The E1 command allows you to menu driven access to data development and remote operations resident in the design adapters and remote operations using P/T adapters.

SYSTEM COMMANDS

The LogicPak offers numerous system commands that allow you to edit and transfer data and set parameters. System commands are accessed by entering a two-character select code from the programmer front panel or a menu indicated code from the terminal. Some commands will prompt for data entry. The system commands and a brief description are listed in the Command Summary table.

The sequence explanations assume no operating errors. If these occur, the programmer signals with a beep and displays a two-digit error code in front panel mode or an error message in terminal mode. It also beeps once when an incorrect key is pressed. Error codes are explained in Appendix A and in your programmer manual. Some errors will return you to the programmer front panel control from the terminal mode.

The entries that you are to make from either the programmer or the terminal are indicated by the entry enclosed in a key symbol. For example:

[ESC]

indicates the ESC (escape) key on the terminal keyboard should be pressed.

When the entry you are to make is variable, appropriate substitutions for the actual values will be used, for example:

[F] [F] [P] [P]

indicates that the appropriate values for the family and pinout code for the device will be entered.

Command Summary

The following definitions describe the display options in the terminal mode. These same function operations are available from the front panel but there will be no terminal display.

Code	Command Name	Front Panel	Terminal	Description of Terminal Mode
E1	Enable Terminal Mode	X		Transfers control of the LogicPak and programmer to the terminal
0	Display Command Menu		X	Causes the LogicPak to redisplay its command menu on the terminal
1	Family and Pinout Code		X	Allows the user to enter the family/pinout code for a logic device
E5 5	Reject Count Option	X	 X	Allows the user to select manufacturers recommended number of pulses or 1 pulse programming
E6 6	Verify Option Option	X	 X	Displays a three item menu for selecting verity and functional test routines
E7 7	Security Fuse Option	X	 X	Displays and allows selecting of the security fuse options
E8 E9 8	Functional Test Data	X X	 X	Enables the functional test data function. Allows you to enter functional test data and perform vector editing.
EA A	Transmit Fuse Pattern	X	 X	Transmits the fuse pattern in the programmer RAM to the serial port
EC C	Transmit JEDEC Data	X	 X	Transmits the contents of the fuse and vector RAM to the serial port in the JEDEC format (see Appendix C)
EB B	Receive JEDEC Data	X	 X	Prepares the programmer to receive JEDEC fuse and vector data from a peripheral device via the serial port

(continued on next page)

Code	Command Name	Front Panel	Terminal	Description of Terminal Mode
ED D	Display Fuse Sumcheck	X	 X	Displays the sumcheck of the fuse data in RAM
EE E	Edit Fuse Pattern	X	 X	Enables the fuse editing function. Fuse states may be changed from blown to unblown or vice-versa in fuse memory.
EF F	Display Config- Number	X	 X	Displays the configuration number of the adapter firmware
CE G	Select Attributes	X	 X	Provides a menu of six attributes, each of which allows you to select one of two options
CTRL Z ESC	Exit Commands		X X	Allows you to exit a programming session and terminate all operations
CTRL Y	Abort Operation		X	Allows you to abort an operation such as fuse map dump, or structured vector error dump
2	Load RAM		X	Loads the programmer with data from a master device
4	Program RAM		X	Programs, verifies and tests a device with programmer RAM data
3	Verify Device		X	Verifies and tests a device

NOTE

These last three programming functions listed in the table are also available from the front panel. The key sequences for front panel operation are in the Getting Started section of this manual and in your programmer manual.

Load RAM with Master Device Data

Before loading the programmer with data from a master device:

1. Place the system in terminal mode by selecting E1 from the front panel of the programmer. The programmer should be connected to the RS-232C serial port of the terminal.
2. Enter the family and pinout code (refer to your P/T adapter User Note) at the terminal, if prompted by the terminal display.

NOTE

If options are desired such as performing a functional test, select options and parameters as needed before proceeding.

Use the following procedure to load the programmer with data from a master device from front panel mode.

Procedure	Front Panel Key Sequence	Front Panel Display
1. Prepare programmer to transfer transfer fuse pattern data from master device to programmer RAM.	COPY DEVICE RAM START	*FAM 00 PIN 00*
2. Accept (or if display is wrong), key in the 4-digit family/pinout code for the device (check the device list in your P/T adapter User Note). (X is the number entered.)	F F P P	*FAM XX PIN ∧ XX*
3. Insert the master device into the socket with the illuminated LED below it.	START	*LOADING DEVICE* ⧄ *LOAD DONE XXXX* (XXXX = sumcheck of data transferred)
4. Remove the master device		

An action symbol ⧄ will be displayed showing the pretesting, programming and verifying of the device. If no errors occur, the terminal displays sumcheck XXXX.

Use the following procedure to load the programmer with data from a master device from the terminal.

Procedure	Terminal Key Sequence	Terminal Display
1. Select the load operation.	2	Command : 2 - Load device Push return to load device
2. Insert the master device into the socket.		Command : 2 - Load device Push return to load device Loading device ... Sumcheck 0000 Command :

An action symbol . . . . will be displayed showing the pretesting, programming and verifying of the device. If no errors occur, the terminal displays sumcheck XXXX.

Program Device with RAM Data

Before programming a device with fuse pattern data previously loaded into the programmer's RAM:

1. Place the system in the terminal mode (select E1 from the front panel).
2. Enter the family and pinout code, if prompted by the terminal.

NOTE

If options are desired such as programming a security fuse, select options and parameters as needed before proceeding.

Use the following procedure the program the device with data in RAM from the front panel mode.

Procedure	Front Panel Key Sequence	Front Panel Display
1. Prepare programmer to transfer fuse data to the blank device.	COPY RAM DEVICE START	*FAM 00 PIN 00*
2. Insert the blank device.	START	*TEST DEVICE* ◻ *PROGRAM DEVICE* ◻ *VERIFY DEVICE* ◻ *PRG DONE 01 XXXX*

NOTE

This step, along with programming the device, verifies that the data was correctly transfered.

3. Remove the device. To program another, insert a blank device and press	START	

An action symbol ◻ will be displayed showing the pretesting, programming and verifying of the device. If no errors occur, the terminal displays sumcheck XXXX.

Use the following procedure to load the programmer with data from a master device from the terminal.

Procedure	Terminal Key Sequence	Terminal Display
1. Select the program operation operation.	4	Command : 4 - Program device Push return to program device
2. Insert the blank device into the socket.	RETURN	Command : 4 - Program device Push return to program device Testing device .... Programming device .... Verifying device .... Sumcheck 0000 Command :

An action symbol . . . . will be displayed showing the pretesting, programming and verifying of the device. If no errors occur, the terminal displays sumcheck XXXX.

Verify and Functionally Test Device

Before verifying and testing a programmed device from terminal control:

1. Place the system in the terminal mode (select E1 from the front panel).
2. Enter the family and pinout code, if prompted by the terminal.

NOTE

If options are desired such as performing a functional test, specifying the number of passes, or choosing the verify modes, select options and parameters as needed before proceeding.

Procedure	Terminal Key Sequence	Terminal Display
1. Insert the device to be verified verified.	3	Command : 3 - Verify device Verifying device .... Sumcheck 0000 Command :

An action symbol . . . . will be displayed showing the verification function under way. Upon completion the terminal will display sum-check XXXX of the device fuses.

Enable Terminal Mode

With the programmer connected to the programmer RS-232C port of the terminal, the terminal will prompt you to enter family codes and pinout codes unless they have already been entered. The terminal will then display the command menu.

Procedure	Front Panel Key Sequence	Front Panel Display
1. Select the terminal mode.	SELECT E 1 START	◻

Display Command Menu

This command causes the PLDS to redisplay its command menu on the terminal, as shown in the previous figure.

Procedure	Terminal Key Sequence	Terminal Display
1. Display command menu on screen.	0	See the following display.

```
Command : 0 - Display menu

DATA I/O CORP. - Programmable Logic Development System - 303A-V04
                     Copyright 1982,1983,1984

     - GENERAL COMMANDS -                      - I/O COMMANDS -
0 - Display menu                          B - Receive JEDEC data
1 - Enter Family/pinout code              C - Transmit JEDEC data
5 - Enter reject count option
6 - Enter verify option
7 - Enter security fuse option
8 - Enter functional test data
F - Configuration number
G - Select attributes

   - DEVICE RELATED  COMMANDS -               - FUSE MAP COMMANDS -
2 - Load device                           A - Display fuse pattern
3 - Verify device                         D - Display fuse sumcheck
4 - Program device                        E - Edit fuse pattern

NOTE - Always transmit an "ESC" before removing adapter
```

Family and Pinout Code

From the programmer front panel control, family and pinout code entry is part of device-related operations (see the Getting Started section in this manual and refer to your programmer manual). This procedure is for operation from the terminal.

NOTE
In the Family and Pinout Code list, find the code numbers corresponding to the device number for the manufacturer of the device. Notice that some devices, such as the AmPAL22V10, have two family/pinout codes. See the P/T adapter user notes for an explanation of the difference.

Procedure	Terminal Key Sequence	Terminal Display
1. Select the option that allows you to enter the family/pinout code.	[1]	`Command : 1 - Enter Family/pinout code` `Family/pinout code 2224`
2. Accept, or if display is incorrect key in the 4-digit family/pinout code for the device you are using.	[F] [F] [P] [P]	NA

NOTE
Space and backspace (CTRL H) may be used to move the cursor back and forth.

Set Reject Count Option

This command allows you to select the number of programming pulses applied to the device fuses before the programmer rejects the device as unprogrammable. The default value of 0 selects the manufacturer's specified number of programming pulses while a value of 1 selects a single programming pulse. Refer to the adapter User Notes for specific entries to select optional reject values.

NOTE

Altera, Cypress, and VTI devices are not supported by this option because of the type of programming algorithms used. The design adapters also do not provide this option.

Use the following procedure to select the reject count option from front panel mode.

Procedure	Front Panel Key Sequence	Front Panel Display
1. Select the reject count option.	SELECT E 5 START	0 0
2. Change the reject count option.	1 START	0 1 **

Use the following procedure to select the reject count option from terminal mode:

Procedure	Terminal Key Sequence	Terminal Display
1. Select the reject count option. (X is the number entered)	5 X	Command : 5 - Enter reject count option Programming reject count options - 0 - Default 1 - Optional Enter option:0 Command :

Select Verify Option

Three options are available for selecting verify and functional test routines. These options are:

Options	Description
0	Default option. Perform fuse verify, followed by structured test (if test vectors are present in RAM), and Logic Fingerprint test (if one or more Logic Fingerprint test cycles are selected), in that order.
1	Perform fuse verify only.
2	Perform structured test and Logic Fingerprint test only, in that order. Does not perform fuse verify.

Option 0 (default) is the option used in normal operation. Option 1 checks the programming of the device fuses without checking device functionality. Use option 2 to functionally test devices with the security fuse blown. In addition, option 2 can be used to learn the Logic Fingerprint test of a device with the security fuse blown. (Fuse data in RAM will be cleared during this operation.) Programming cannot occur with option 2 selected.

Verify options must be entered from either the programmer's keyboard or a terminal. The option will remain in effect until it is changed or until the unit is powered down. To reselect the default, key in option 0.

Use the following procedure select the functional test option from the front panel mode:

Procedure	Front Panel Key Sequence	Front Panel Display
1. Select the verify options.	SELECT E 6 START	0 0

As an example, from the options described above

2. Select functional test.	2 START	0 2	**

Use the following procedure to select the functional test option from the terminal.

Procedure	Terminal Key Sequence	Terminal Display
1. Select the verify options.	6 2	See the following display.

```
Command : 6 - Enter verify option

0 - Sequence - fuse verify, structured test, Logic Fingerprint
1 - Fuse verify only
2 - Sequence - structured test, Logic Fingerprint

Enter verify option: 2
 Command :
```

Select Security Fuse Option

Some logic devices are equipped with protective fuses called security fuses. Once the security fuses are programmed, the fuse states in the logic array cannot be copied. Programming the security fuses makes it very difficult to pirate a device design.

NOTE

Refer to your P/T adapter User Note for adapter specific security fuse information.

With the LogicPak you can either enable programming of the security fuse at all times, allow programming only when security fuse data are downloaded to the PLDS via the serial port, or disable programming completely, whether security fuse data are downloaded or not.

When the security fuse has been blown, a Logic Fingerprint test and structured test can still be performed, but a fuse verity operation is not possible.

To enable programming of security fuses, two conditions must be met: 1) the security fuse state in the programmer RAM must be 1 (or true), and 2) security fuse programming must be enabled. Once the security fuse option is selected, it will remain in effect until changed or until the programmer is turned off.

When security fuse data are entered into RAM in the JEDEC ASCII-logic format, data in the G field indicate the state of the security fuse. The G field does not affect the enable state of the security fuse option. The enable state must be entered separately. This can be done before or after loading JEDEC ASCII-logic format data.

Security fuse states cannot be loaded from a master device.

CAUTION

Once the security fuse is blown, you cannot verify the state of any fuse in the device. For devices which are not erasable the process cannot be reversed; therefore, be certain that you want to program the security fuse before you activate this function. Attempting to reprogram the device after the security fuse is blown will alter the original fuse pattern and render the device inoperative.

Security Fuse Select-Code Options:

Option	Description
0	Default option. Disable programming and set the security fuse state in RAM to 0 (unprogrammed).
1	Disable programming and set security fuse state in RAM to 1 (programmed).
2	Enable programming and set security fuse state in RAM to 0. (Data downloaded in the JEDEC format can change the security fuse state to 1.)
3	Enable programming and set security fuse state in RAM to 1. (Data downloaded in the JEDEC format can change the security fuse state back to 0.)

Use the following procedure to select the security fuse option from front panel mode.

Procedure	Front Panel Key Sequence	Front Panel Display
1. Select security fuse option.	SELECT E 7 START	0 0

As an example, use the following procedure to enable security fuse programming and set security fuse state in RAM to 1 (option 3).

Procedure	Front Panel Key Sequence	Front Panel Display
1. Set security fuse state to 1.	3 START	0 3 **

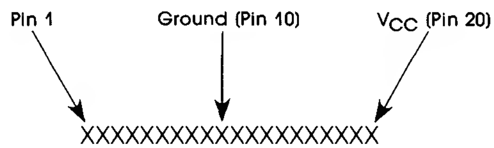
Procedure	Terminal Key Sequence	Terminal Display
1. Select security fuse option.	7 3	See below

```
Command : 7 - Enter security fuse option

           SECURITY      SECURITY FUSE
OPTION     FUSE DATA     PROGRAMMING

   0 -------- 0 -------- disabled
   1 -------- 1 -------- disabled
   2 -------- 0 --------- enabled
   3 -------- 1 --------- enabled

Enter Security Fuse option: 3
 Command :
```

Enter Functional Test Data

Functional test data includes information for the Logic Fingerprint test and also the test vectors used by P/T adapters for testing of a programmed device. The Logic Fingerprint test information consists of three components:

- ☐ The number of test cycles to be performed during the Logic Fingerprint test (described in Appendix B). The default value is 00, which disables the Logic Fingerprint test.
- ☐ The Logic Fingerprint starting vector. This is an arbitrary binary sequence, each bit of which corresponds to a pin on the device under test. The starting vector format for a 20-pin device is shown below. Each "X" represents a "1" or a "0" to apply a logic high or logic low to the corresponding pin. The default value is all 0's. Values entered for V_{CC} and ground have no effect on the device under test.



- ☐ The starting vector is used to initialize the Logic Fingerprint, and is one of the components (along with the device type, number of test cycles, and fuse pattern) which determine the resulting Logic Fingerprint test signature. Note that different Logic Fingerprint test signatures may result for a given logic design, depending on the choice of starting vector.
- ☐ The Logic Fingerprint test signature itself is the result of performing the Logic Fingerprint test, as described later in this section.

Logic Fingerprint test data may be entered from either the front panel or the terminal. From the front panel, the number of test cycles and the starting vector for the Logic Fingerprint test may be entered, and the resulting Logic Fingerprint test signature may be viewed or entered. Structured test vectors may not be entered or edited from the front panel but only from a terminal or serial download. All functional test data may be entered from the terminal, including number of test cycles, starting vector, the Logic Fingerprint test signature itself, and the test vectors.

NOTE

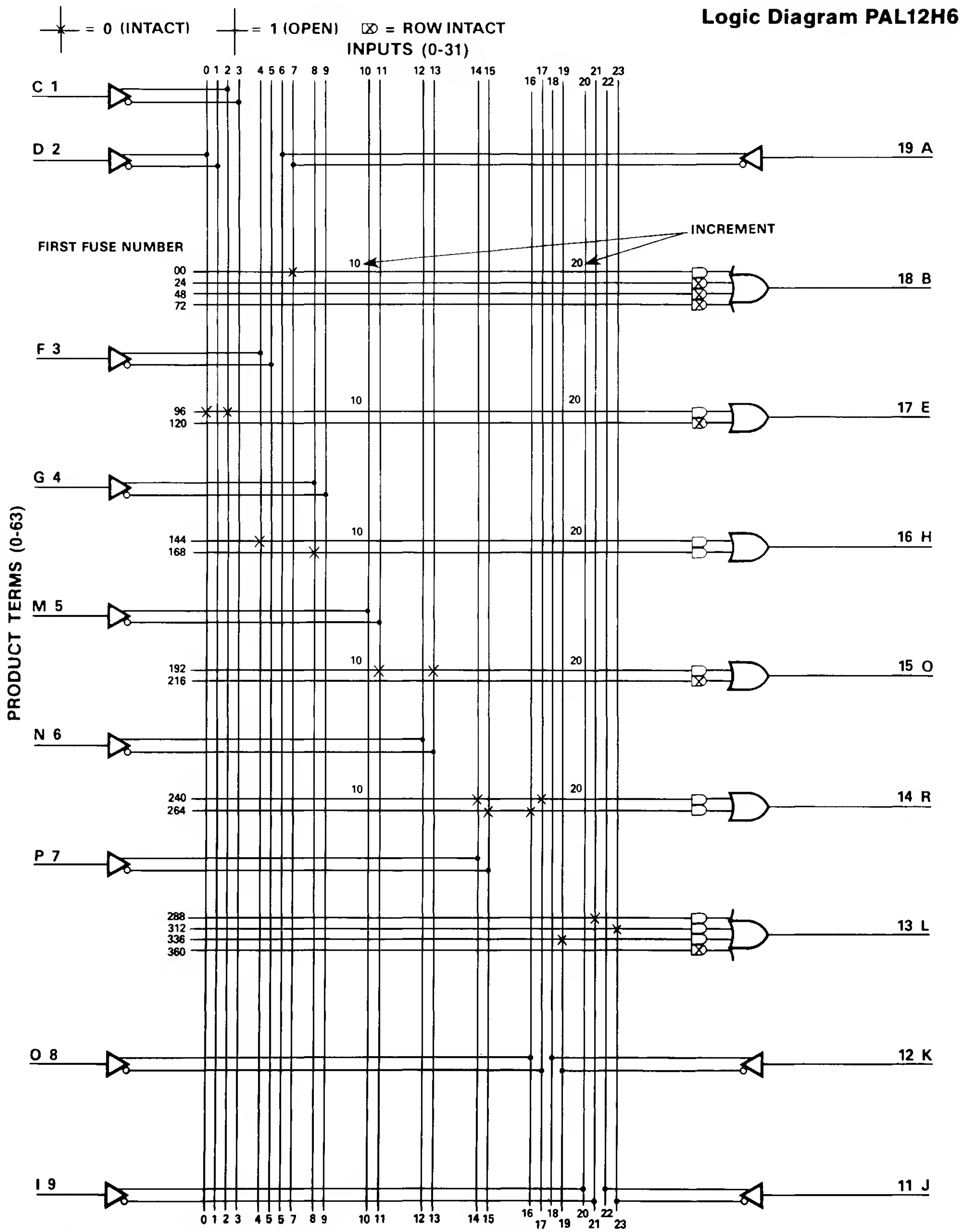
If a value is entered for the Logic Fingerprint test signature, it should be either O O O O O O O O or a known-good value corresponding to the number of test cycles, starting vector, device, and fuse patterns under test. A value of O O O O O O O O will cause the LogicPak to learn the correct Logic Fingerprint test signature when a Load, Program, or Verify operation is performed. When in Load, the correct Logic Fingerprint will be learned independently of the value entered.

If "Device Selection Error" (Error 30) appears when you select functional test data, you must specify family code and pinout code to define the starting vector width.

In the subsections which follow, functional test data will be entered to test the Basic Gates design example[a]. This figure shows an example of one method of programming a PAL. Structured vectors, fuse map display, JEDEC file, and sum-check are functions that are illustrated in the Basic Gates example[a].

[a]Adapted from the MMI PAL HANDBOOK, available from Monolithic Memories, Inc., 1165 Arques Avenue, Sunnyvale, California 94086.

Logic Diagram PAL12H6



Adapted from MMI PAL Handbook

From the front panel the number of test cycles and the Logic Fingerprint starting vector may be entered, and the Logic Fingerprint test signature may be viewed or entered. Use the following procedure to set the number of Logic Fingerprint Test cycles.

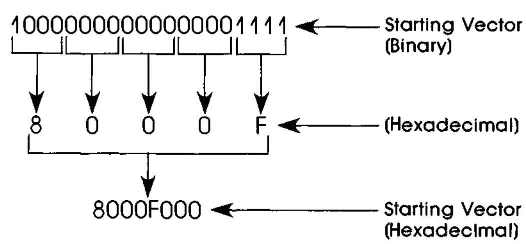
Procedure	Front Panel Key Sequence	Front Panel Display
1. Select the function to set the number of Logic Fingerprint Test cycles.	[SELECT] [E] [8] [START]	*0 0*

As an example, use the following procedure to enable one cycle of testing.

Procedure	Front Panel Key Sequence	Front Panel Display
1. Enable one cycle of testing.	[0] [1] [START]	*0 1* **

Logic Fingerprint and Starting Vector

The starting vector must be converted from the binary form to hexadecimal for entry from the front panel. For our Basic Gates example, we will choose an arbitrary test vector as shown:



Use the following procedure to select the Logic Fingerprint and starting vector function. The unused portion of the 32-bit vector is assumed to be zeroes and must be included in the hexadecimal vector entry.

Procedure	Front Panel Key Sequence	Front Panel Display
1. Select the starting vector function.	[SELECT] [E] [9] [START]	*0000 B1*

NOTE
The eight-character starting vector is entered into the programmer in two fields. B1 identifies the first field.

Procedure	Front Panel Key Sequence	Front Panel Display
2. Enter the first four hexadecimal digits.	[8] [0] [0] [0]	*8000 B1*
3. Select the B2 field.	[START]	*0000 B2*

NOTE
B2 represents the second field.

Procedure	Front Panel Key Sequence	Front Panel Display
4. Enter the remaining hexa-decimal digit.	[F] [0] [0] [0]	*F000 B2*
5. Display the starting vector.	[START]	

NOTE
The zeros are ignored, but are needed to correctly position the "F". This vector, when applied to the Basic Gates example, produces the Logic Fingerprint test signature: ED37A9E4 (hexadecimal).

Procedure	Front Panel Key Sequence	Front Panel Display
6. Enter the fingerprint test.	[E] [D] [3] [7]	*ED37 E1*

NOTE
The first four characters are displayed as the E1 field.

Procedure	Front Panel Key Sequence	Front Panel Display
7. View the E2 field.	[START]	
8. Enter the values for the E2 field.	[A] [9] [E] [4]	*A9E4 E2*
9. Display the E2 field.	[START]	*A9E4 E2* **

Use the following procedure to select the Logic Fingerprint and starting vector from the terminal.

Procedure	Terminal Key Sequence	Terminal Display
1. Select the function to enter functional test data.	8	See the following display.

```
Command : 8 - Enter functional test data

Cycles for Fingerprint: 01
Fingerprint starting vector: 10000000000000001111
Fingerprint: ED37A9E4
```

As each prompt appears, you may modify the current values using the following steps:

1. Move the cursor forward (using the spacebar) and backward (using the backspace) along the displayed value until it is positioned over the symbol to be changed.
2. Press the desired symbol.
3. Enter RETURN at any point to move to the next prompt.
4. CTRL Z is used to exit the functional test data entry mode.

Vector Editing

Vectors are created in RAM by downloading JEDEC "V" fields, simulating a source file containing a function table (using PALASM), or by using the vector editor in the terminal mode. (Function 8 on the menu.)

When the Logic Fingerprint test information has been entered (or skipped by entering RETURN), the vector editor menu appears (see following example), and a prompt appears for the vector number to be edited. The default vector is 0001, as shown in the next example.

NOTE

Some P/T adapters support both the dual-in-line (DIP) package and the chip carrier package. Generally, the chip carrier has more pins than the DIP socket. Structured vectors must be written for the DIP socket and the pinout is automatically carried across to the functionally identical pinout on the chip carrier.

```
 Command : 8 - Enter functional test data

Cycles for Fingerprint: 01
Fingerprint starting vector: 10000000000000001111
Fingerprint: ED37A9E4

           - DISPLAY  -                             - EDITING COMMANDS -
0 ------------- Display menu           D ------- Delete (Kill) current vector
Return -------- Go to next vector      A ------- Repeat current vector
U ------------- Up (previous vector)   CTRL Z -- Exit vector editor
#<N> ---------- Go to vector <N>
Space --------- Move cursor right
BKSP (CTRL H) - Move cursor left
CTRL Z -------- Exit vector editor

Edit structured vector:  0000
0001:   1100XX10XNXXXHXLHH0N
0002:   --------------------
```

The vector editor is a fixed-format line editor with the first column of the displayed line reserved for command characters, as shown in the next example.

A character entered in the first column (normally blank) is interpreted as a command and acted upon immediately; otherwise, vector editing is not processed until a RETURN is entered (at any point on the line). The command characters recognized in the first column are 0, U, # , D, and R; see the table for command character definitions.

Command	Description	Activity
0 (zero)	Display menu	Redisplays menu and restarts editing on the same vector.
U	Up (previous vector)	Moves editing to the next lower vector number (the vector one 'up' on the screen).
# (N)	Go to vector (N)	Entering a # in the command column causes the vector editor to prompt for the desired vector number (default = 0001). Entering a vector number greater than the last vector will move you to the last vector.
D	Delete current vector	Current vector is deleted, and all higher vectors moved down one. Current vector number is redisplayed with new vector.
R	Repeat current vector	Creates a copy of the current vector immediately following the current vector. The copy is displayed, with its vector number (one greater than the original). This command may be given for any vector, and existing vectors will be moved to accommodate the new copy.

During operation, the vector editor copies the selected vector to a temporary buffer where all editing changes are made. Then, when a RETURN command is entered, the temporary buffer is examined for legal characters before copying back to vector memory. You are not allowed to proceed to another vector until all characters are legal in the current vector. Typing a CTRL Z to exit the vector editor will leave the selected vector in its original state. An *empty vector* is represented by a dash in all pin positions. This will appear as the first vector in an empty vector editor buffer, or as one past the last vector where data are present in memory. All vectors are numbered lower than the empty vector.

Use the following procedure to edit a vector.

Procedure	Keystrokes	Definition
1. Increment (spacebar) or decrement (backspace) by one position.	spacebar backspace	Moves the cursor forward or backward along the test vector.
2. Select the desired test conditions to enter into the vector image from the allowable characters.	0	Drive input low.
	1	Drive input high.
	2-9	Drive input to super-voltage #2-9.
	C	Drive input low, high, low.
	K	Drive input high, low, high.
	N	Power pins and outputs not tested.
	L	Test output low.
	H	Test output high.
	Z	Test output for high impedence.
	F	Float input or output.
	•X	Ignore input or output (not JEDEC format).
	P	Preload (applied to clock pin).

*X is not defined in the JEDEC format. The X is treated as an N for outputs and leaves an input at its previously defined state.

NOTE

Test conditions 2 through 9 specify non-TTL levels (supervoltages) that access special device features. A device may be damaged by improper use of supervoltages.

Procedure	Keystrokes	Definition
3. Move the cursor to the next vector or exit the editor.	RETURN	At any point moves the cursor to the next vector.
	CTRL Z	Exits the vector editor.

Register Preload

In some registered logic devices, the internal registers can be arbitrarily loaded to a desired state. This capability allows easier functional testing by providing a means of achieving states which may be difficult or impossible to enter by normal state transitions.

For devices which have the register preload feature preload is accomplished by using a "preload vector", a structured vector which has a "P" symbol in the clock pin position. Also in the preload vector are special symbols in the positions of the pins associated with loading of the registers. The symbols used in the preload vector and their functions are described in the following table.

Preload Vector Symbols

P	Identifies preload vector and invokes preload algorithm. (Allowed on clock pin only, otherwise treated as "X".)
O	Preloads a logic "0" into the register $\overline{Q}$ output, meaning a logic "1" wil be loaded into the register Q output. Does not test device outputs.
1	Preloads a logic 1 into the register $\overline{Q}$ output, meaning a logic O will be loaded into the register Q output. Does not test device outputs.
L	Preloads register with the appropriate level such that a logic O appears on the device output pin. Also tests the preloaded device output and indicates an error if a logic O is not found. Not allowed for some devices.
H	Preloads register with the appropriate level such that a logic 1 appears on the device output pin. Also tests the preloaded device output and indicates an error if a logic 1 is not found. Not allowed for some devices.

For adapter specific information on devices which have the preload feature, refer to your adapter User Note.

All pins not used in the device's preload algorithm (regardless of the symbol placed in the preload vector pin position) are treated as "X"s (left in their previous state). Pins which are used in the preload algorithm may not return to their original state following preload. For example, to preload a 20-pin device with preload pins (most likely device outputs) 12 through 19, you might apply the following preload vector: (clock pin assumed to be pin 1).

0001: PXXXXXXXXNXHLHLHLHLN

When the preload vector is applied during functional testing, the device-specific preload algorithm is invoked and the registers are loaded with the appropriate data to make the outputs high "H" or low "L". The output pins are then tested to verify that the preload was successful.

Assuming the device has an inverter between the register output and the output pin, another method of achieving the same results as above is to use the following two vectors:

0001: PXXXXXXXXNX10101010N

0002: XXXXXXXXXN0HLHLHLHLN

The first vector is a preload vector using "1"s and "0"s to load the Q output of the register with the data indicated (thus making the $\overline{Q}$ outputs of the registers the complements of the data in the vector). Since we have assumed an inverter between the $\overline{Q}$ output and the output pin, the data found on the output pins after execution of the preload vector should reflect the "1"s and "0"s in the preload vector. The second vector shown is a conventional structured vector which tests the outputs for the desired data.

The "1" and "0" preload symbols are most useful for preloading registers whose state cannot be read at a device pin, or for any case in which the user is concerned with setting up the state of the registers not necessarily the state of the output pins.

The H&L preload symbols are used to preload the states of output pins whose states are determined by the data in internal registers. The P/T adapter firmware determines what data should be placed in the internal registers to provide the correct outputs. Users concerned with preloading the state of the internal registers can use the H&L preload vector to load and automatically verify internal register states provided that data inversion (if any) between registers and outputs is considered. Some devices depend upon a fuse state to determine the data inversion (if any) between the registers and the outputs. If this is the case, the H&L style preload vector is not allowed and an error will result.

Display and Transmit Fuse Pattern

This command transmits the fuse pattern in the programmer data RAM to the serial port. The fuse states may be shown as a series of "1"s and "0"s or a series of "-"s and "X"s; see the subsection on selecting characters. The "1" or "-" represents a high-resistance or "blown" fuse in a fuse link device. The "0" or "X" represents a low-resistance or "intact" fuse. Each fuse can be identified by a decimal fuse number, as shown in the figure. The fuse states are arranged in a matrix that corresponds to the logic diagram of the device (see the figure for the Logic Diagram for Basic Gates Example). This is useful for comparing or copying a displayed fuse pattern to the device logic diagram.

```
 Command : A - Display fuse pattern

      00          10
0000  XXXXX---XX  -XX--XX-XX
0020  XXXXX-----  -----XXXXX
0040  ---------X  XXXXXXXX--
0060  XXX----XXX  X----XXXXX
0080  ----------  -----XXXXX
0100  XXXX------  ----------
0120  --------X-  ---------X
0140  XXXXXXXXXX  XXXXXXX---
0160  XXXXXXXXXX  XXXXXXXXXX
0180  ------XXX-  --------XX
0200  XXXXXXXXXX  XX-X-X-X-X
0220  XXXXXXXXXX  X-----XXXX
0240  -------X-X  XXX--XXXXX
0260  -----XXX--  --XXXX--XX
0280  XXXXXXXXXX  -XXXXXXXXX
0300  XXXXXXXXXX  XXXXXXXXXX
Sumcheck 10B0

 Command :

 Note: - = open
       X = intact

 Fuse Number = First Fuse Number + Increment
```

NOTE

Sending certain control characters to the PLDS during the course of fuse pattern display will affect the display. The output may be stopped by sending a CONTROL S (DC1 or ASCII, 11 hex) and then restarted by sending a CONTROL Q (DC3 or ASCII 13 hex).

A CONTROL Y (ASCII, 19 hex) will terminate the transmission and return to the terminal or front panel operation.

An ESC (escape) character (ASCII 1B hex) will also terminate the transmission and return to front panel operation.

If the underblow/overblow attribute is enabled, the fuse map display may contain some U's (underblow) or B's (overblow). See the subsection on Select Attributes for the definition of underblow and overblow.

The last character of the fuse pattern transmission is either CONTROL C (ETX or ASCII 03) or a CONTROL Z (ASCII 1A hex). (See the section on Select Attributes.)

Use the following procedure to display the fuse pattern on the terminal screen from front panel mode.

Procedure	Front Panel Key Sequence	Front Panel Display
1. Display the fuse pattern.	[SELECT] [E] [A] [START]	*XXXX* ** ⍰

NOTE
⍰ is the action symbol. XXXX Is the fuse data checksum.

Use the following procedure to display the fuse pattern on the terminal screen from the terminal.

Procedure	Terminal Key Sequence	Terminal Display
1. Display the fuse pattern	[A]	Command : A - Display fuse pattern

```
        00          10
0000  XXXXX---XX  -XX--XX-XX
0020  XXXXX-----  -----XXXXX
0040  ---------X  XXXXXXXX--
0060  XXX----XXX  X----XXXXX
0080  ----------  -----XXXXX
0100  XXXX------  ----------
0120  --------X-  ---------X
0140  XXXXXXXXXX  XXXXXXX---
0160  XXXXXXXXXX  XXXXXXXXXX
0180  ------XXX-  --------XX
0200  XXXXXXXXXX  XX-X-X-X-X
0220  XXXXXXXXXX  X-----XXXX
0240  -------X-X  XXX--XXXXX
0260  -----XXX--  --XXXX--XX
0280  XXXXXXXXXX  -XXXXXXXXX
0300  XXXXXXXXXX  XXXXXXXXXX
Sumcheck  10B0
```

JEDEC Format Data Exchange

Transmit JEDEC Data

This command transmits the contents of the fuse and vector RAM to the serial port in the JEDEC format (see Appendix C).

The following characteristics apply to JEDEC transmission:

- The output may be halted by sending a CONTROL S (DC1 ASCII, 11 hex) and restarted by sending a CONTROL Q (DC3 or ASCII, 13 hex).
- An ESC character (ASCII, 1B hex) will abort the transmission and return to the programmer front panel operation.
- A CONTROL Y (ASCII, 19 hex) will terminate the transmission and return to the terminal or programmer front panel operation.
- The Logic Fingerprint test fields (S, R, and T) are not sent if the number of cycles is 0.
- The G field is sent only if security fuse data is a *1*.
- The fuse checksum (C field) is the 16-bit sum of all fuse states (i.e., from fuse 0 to the fuse limit for the device). See the following example.

```
<STX>*F0*L0000
01001110 00001000 11110000 11111111 01010001*
C021A*
<ETX>0000

The F0* cleared all the fuse RAM to 0. The L field transmitted 40
fuse states starting at 0.

Fuse number 00 01 02 03 04 05 06 07  08 09 10 11 12 13 14 15  16
State        0  1  0  0  1  1  1  0   0  0  0  0  1  0  0  0   1

                MSB             LSB

                7 6 5 4 3 2 1 0

   0000         0 1 1 1 0 0 1 0          72
   0008         0 0 0 1 0 0 0 0          10
   0016         0 0 0 0 1 1 1 1          0F
   0024         1 1 1 1 1 1 1 1          FF
   0032         1 0 0 0 1 0 1 0          8A
   0040         0 0 0 0 0 0 0 0          00
   0048         0 0 0 0 0 0 0 0          00
   xxxx         0 0 0 0 0 0 0 0          00
                                        ----
                                        021A
```

Use the following procedure to transmit JEDEC data from front panel mode.

Procedure	Front Panel Key Sequence	Front Panel Display
1. Select the transmit data function.	SELECT E C START	XXXX ** ⍁

NOTE
⍁ is the action symbol. XXXX is the fuse data sumcheck.

Use the following procedure to transmit JEDEC data from the terminal.

Procedure	Terminal Key Sequence	Terminal Display
1. Select the transmit data function.	C	See the following display.

```
 Command : C - Transmit JEDEC data
PAL12H6           PAL DESIGN SPECIFICATION
P7000              JANE ENGINEER    5-21-83
BASIC GATES
DATA I/O
*
QP20*
QF0384*
L0000
111111101111111111111111
000000000000000000000000
000000000000000000000000
000000000000000000000000
010111111111111111111111
000000000000000000000000
111101111111111111111111
111111110111111111111111
111111111110101111111111
000000000000000000000000*
L0240
111111111111110110111111
111111111111111001111111
111111111111111111111011
111111111111111111111110
111111111111111111101111
000000000000000000000000*
V0001 XXXXXXXXXNXXXXXXXH0N*
V0002 XXXXXXXXXNXXXXXXXL1N*
V0003 00XXXXXXXNXXXXXXLXXN*
V0004 01XXXXXXXNXXXXXXLXXN*
V0005 10XXXXXXXNXXXXXXLXXN*
V0006 11XXXXXXXNXXXXXXHXXN*
V0007 XX00XXXXXNXXXXXLXXXN*
V0008 XX01XXXXXNXXXXXHXXXN*
V0009 XX10XXXXXNXXXXXHXXXN*
V0010 XX11XXXXXNXXXXXHXXXN*
V0011 XXXXXXXX0N00HXXXXXXN*
V0012 XXXXXXXX0N01HXXXXXXN*
V0013 XXXXXXXX0N10HXXXXXXN*
V0014 XXXXXXXX1N00HXXXXXXN*
V0015 XXXXXXXX1N11LXXXXXXN*
V0016 XXXX00XXXNXXXXHXXXXN*
V0017 XXXX01XXXNXXXXLXXXXN*
V0018 XXXX10XXXNXXXXLXXXXN*
V0019 XXXX11XXXNXXXXLXXXXN*
V0020 XXXXXX00XNXXXLXXXXXN*
V0021 XXXXXX01XNXXXHXXXXXN*
V0022 XXXXXX10XNXXXHXXXXXN*
V0023 XXXXXX11XNXXXLXXXXXN*
T01*
S00000000000000000000*
RAB472BBB*

C1BB9*
2642
```

Receive JEDEC Data

This command prepares the programmer to receive JEDEC formatted fuse and vector data from a peripheral device via the serial port.

NOTE
The D field is ignored by the translator. The correct family and pinout code must be entered before receiving JEDEC data.

Three types of errors may be caused by receiving improper data in the JEDEC format (see the next table).

Translator Input Error Codes

Error	Description	Possible Fields
82	SUMCHK ERR	Transmission checksum
84	INVALID DATA	EXT, F, L, S, V
91	I/O FORM ERR	C, G, L, P, R, T, V

You may determine the field in which the error occurred by examining data RAM location 0408; the ASCII value (hexadecimal) of the field is stored here (see the following table). More information about the possible cause of the error may be found in the table on Translator Input Error Codes

Field Identifier	ASCII Character Hex Value
(EXT)	03
C	43
F	46
G	47
L	4C
P	50
QF/QP	51
R	52
S	53
T	54
V	56

Use the following procedure to examine the data RAM location 0408 from front panel mode.

Procedure	Front Panel Key Sequence	Front Panel Display
1. View the RAM address.	EDIT	*EDIT ADDR XXXX*
	4 0 8 START	*0408 D**∧RXX*

NOTE
XXXX is the current address. XX is the field identifier in hexadecimal.

The transmission checksum computed by the LogicPak may be found by examining data RAM locations 405 and 406 in a similar manner.

Error	Display	Field	Possible Cause
82	SUMCHK ERR	EXT	Transmission checksum of all ASCII characters does not match the computed value.
84	INVALID DATA	EXT	Fuse sumcheck does not match computed sumcheck. The comparison is not made until the transmission is complete, so the field is stored as EXT rather than C. The sumcheck includes the entire fuse RAM as defined by the family and pinout code, not just the fuse states sent.
		F	Invalid character in field. Only *1* and *O* are allowed.
		L	A space or carraige return did not follow the fuse number.
		L	An invalid character was in the fuse state field. Only *1* and *O* are allowed. Spaces, line feeds, and carriage returns are ignored.
		S	Invalid character in field. Only *1*, *O*, and *N* are allowed.
		V	Too few or too many test conditions.
91	I/O FORM ERROR	C	Invalid character in field, must be 4 digit hexadecimal number.
		G	Invalid character in field. Only *1* or *O* are allowed.
		L	Fuse number exceeds fuse limit for device or invalid fuse number (must be decimal number).
		P	Too few or too many pins or invalid pin number for device.
		T	Test cycles greater than 99.
		R	Invalid character in field: must be 8-digit hexadecimal number.

Use the following procedure to receive JEDEC data from front panel mode.

Procedure	Front Panel Key Sequence	Front Panel Display
1. Select the recieve JEDEC data function.	SELECT E B START	XXXX ** ⌷

NOTE
⌷ is the action symbol. XXXX is the fuse array sumcheck.

Use the following procedure to receive JEDEC data from the terminal.

Procedure	Terminal Key Sequence	Terminal Display
1. Select the receive JEDEC data function.	B	Command : B - Receive JEDEC data

Edit Fuse Pattern

The individual fuses that make up a PLD fuse map may be edited in RAM using the fuse map editor. Fuse states may be changed from blown to unblown or vice-versa on a downloaded fuse map, a fuse map generated by assembly of source code, or loaded from a device.

Fuse states may be edited one at a time from the front panel, or in line editor fashion from a terminal. In the examples that follow, assume that we are editing the Basic Gates fuse map, representing the logic diagram as shown previously. If "Device Selection Error" appears when you enter the fuse editor, you must specify the family and pinout code to define the fuse map.

Use the following procedure to enter the fuse editor from the front panel mode.

Procedure	Front Panel Key Sequence	Front Panel Display
1. Enter the fuse editor.	[SELECT] [E] [E] [START] [X] [X] [X] [X] [START]	*XXXX* **

XXXX is the decimal number of fuse being edited; ** is binary state of fuse number (00 or 01).

You can scroll to the desired fuse number by using the START and REVIEW keys, or specified directly by entering the fuse number XXXX, as shown above. The data displayed on the right reflects the current state of the selected fuse:

01 = high-resistance, *blown* fuse
00 = low-resistance, fuse intact

Entering a "0" or a "1" while displaying a selected fuse will store that state for the fuse.

Use the following procedure to prepare the programmer and change fuse number 98 in our Basic Gates example from unblown to blown from front panel mode.

Procedure	Front Panel Key Sequence	Front Panel Display
1. Prepare programmer to change fuse number 98 to the blown state.	[SELECT] [E] [E] [START]	*0000*
2. Enter the decimal fuse number.	[9] [8] [START]	*0098 00*

This display indicates that RAM data for fuse 98 is set for *don't program*.

Procedure	Front Panel Key Sequence	Front Panel Display
3. Change fuse number 98 to the blown state.	[1] [START]	*0099 01*

(Fuse number increments automatically.)

Procedure	Front Panel Key Sequence	Front Panel Display
3a. Decrement from fuse number 99 to 98.	[REVIEW]	*0098 01*

Use the following procedure to change fuse number 98 in our Basic Gates example from unblown to blown from the terminal.

Procedure	Terminal Key Sequence	Terminal Display
1. Enter the fuse editor.	E	See the following display.

```
Command : E - Edit fuse pattern

      - COMMANDS -                           - FUSE EDITING -
0 ---------------- Display menu        Space ------------ Move cursor right
# ---------------- Go to fuse number   BKSP (CTRL H) ---- Move cursor left
CTRL Z ----------- Exit fuse editor    Return ----------- Edit next row
                                       CTRL Z ----------- Exit fuse editor

Enter decimal fuse number :  0000
```

Procedure	Terminal Key Sequence	Terminal Display
2. Specify a fuse number directly, or display the first fuse row.	9 8 RETURN	Read the following description and refer to the next display.

The fuse editor is a fixed-format line editor. Refer to the following figure for the description of the editing functions. Once the display is on the screen, you can begin editing.

The specified fuse numbers are shown on the far left column of the display. Any fuse number may be specified, regardless of row boundaries, and the display will follow this convention.

When the display first appears, the cursor will be positioned in the first column of the first line of the fuse editor display. This column is reserved for the following command characters only: 0 (zero) and #. When either of these two characters are entered in this column, the command is performed immediately without pressing the RETURN key.

0: When a 0 is entered, the opening menu will be displayed and to resume editing, you must press E again to re-enter the fuse editor.

#: After you press the # key, you will be prompted to enter another fuse address or to press RETURN without entering another address, to resume editing. This allows you to move throughout the data and speeds up editing by taking you directly to the fuses you want to change.

The fuse editor display shows *N* consecutive fuses, where *N* is the number of fuses in one row of the selected device. Entering a RETURN at any time moves the editor to the fuse one row down from the previously specified fuse. Index marks are shown over every tenth fuse in the row displayed, for easy location of fuses. Select Code CE from the programmer keyboard or G from the terminal keyboard can change the display of blown and unblown fuses from the default of X/- to 0/1. See the section on Select Attributes.

The fuse editor copies the selected row to a temporary buffer where all editing changes are made. Then, when a command or RETURN is entered, the editing buffer is examined for legal characters before copying back to the fuse map. You are not allowed to proceed to another row until all characters are legal in the current row. Typing a CTRL Z to exit the fuse editor from an untested edited row will leave the row in its original state.

```
 Command : E - Edit fuse pattern

        - COMMANDS -                              - FUSE EDITING -
0 ---------------- Display menu          Space ------------ Move cursor right
# ---------------- Go to fuse number     BKSP (CTRL H) ---- Move cursor left
CTRL Z ----------- Exit fuse editor      Return ----------- Edit next row
                                         CTRL Z ----------- Exit fuse editor

Enter decimal fuse number :  0000
        !         !         !
0000    XXXXXXXXXXXXXXXXXXXXXXXX
        !         !         !
0024    XXXXXXXXXXXXXXXXXXXXXXXX
        !         !         !
0048    XXXXXXXXXXXXXXXXXXXXXXXX
        !         !         !
0072    XXXXXXXXXXXXXXXXXXXXXXXX
        !         !         !
0096    XXX---------------------
        !         !         !
0120    XXXXXXXXXXXXXXXXXXXXXXXX
        !         !         !  <---- Mark Every Tenth Fuse
0144    ----X-------------------
└──┘   └┘└──────────────────────┘
 ^      ^           ^
 |      |           └------------ Fuses in Row
 |      └------------------------ Command Character
 └------------------------------- First Fuse Number
```

Use the following procedure to edit a fuse row from the terminal:

Procedure	Terminal Key Sequence	Terminal Display
1. Move the cursor back and forth along the displayed row until it is positioned over the fuse to be changed.	SPACEBAR BACKSPACE	See the following display.
2. Select the desired symbol to represent the fuse state from those in the next column.	1 0 - X	See the following display.
3. Display the menu or move to to a specific fuse number by entering either a 0 (zero) or #.	0 #	See the following display.
4. Move to the next row.	RETURN	See the following display.
5. Exit the fuse editor.	CTRL Z	See the following display.

```
Command : E - Edit fuse pattern

        - COMMANDS -                                  - FUSE EDITING -
0 ---------------- Display menu            Space ------------ Move cursor right
# ---------------- Go to fuse number       BKSP (CTRL H) ---- Move cursor left
CTRL Z ----------- Exit fuse editor        Return ----------- Edit next row
                                           CTRL Z ----------- Exit fuse editor

Enter decimal fuse number :  0000
       !         !         !
0000   XXXXXXXXXXXXXXXXXXXXXXXX
       !         !         !
0024   XXXXXXXXXXXXXXXXXXXXXXXX
       !         !         !
0048   XXXXXXXXXXXXXXXXXXXXXXXX
       !         !         !
0072   XXXXXXXXXXXXXXXXXXXXXXXX
       !         !         !
0096   XXX---------------------
       !         !         !
0120   XXXXXXXXXXXXXXXXXXXXXXXX
       !         !         !
0144   ----X-------------------
```

Editing fuse number 98 as demonstrated in the following figure, may be done in two ways. As one method, you can enter the fuse editor and type RETURN until the desired row appears (beginning with fuse 0096), resulting in a display that matches the device data sheet, and then space three times to locate fuse 98. The display in this case will resemble that in the following example.

```
Enter decimal fuse number :  0098
       !         !         !
0098   X---------------------XX
       !         !         !
0122   XXXXXXXXXXXXXXXXXXXXXX--
       !         !         !
0146   --X---------------------
       !         !         !
0170   ------X-----------------
```

Alternatively, fuse number 98 may be directly specified. When this is done, a fuse *row* is displayed which begins with fuse number 0098 and does not match any of the rows in the logic diagram shown previously. Fuse number 98 may now be modified without counting spaces, and subsequent RETURNs will jump to the fuses directly below fuse 98 in the same column (122, 146, 170, etc.). The preceding example shows the display when this method is used.

Display Configuration Number

Use the following procedure to display the configuration number of the adapter firmware from front panel mode. Configuration numbers are used as serial numbers for firmware.

Procedure	Front Panel Key Sequence	Front Panel Display
1. Display configuration number.	SELECT E F START	XXXX **

Use the following procedure to display the configuration number of the adapter firmware from the terminal. Configuration numbers are used as serial numbers for firmware.

Procedure	Terminal Key Sequence	Terminal Display
1. Display configuration number.	F	Command : F - Configuration number XXXX Command :

NOTE
XXXX is the configuration number of the firmware in the adapter plugged into the LogicPak.

Select Attributes

This command allows you to select one of two options for any of six attributes, listed below. The only options available to the PALASM and H&L design adapters are those numbered 0 thru 7:

Option	Description
0	Echo (full duplex): PLDS echoes all characters received at the serial port.
1	No echo (half duplex).

NOTE

The default echo mode will depend upon the programmer being used. The Model 29 and 100A programmers will power up in the "no echo" mode, while the Model 19 will power up in the "echo" mode.

Option	Description
2	JEDEC full mode: described by the JEDEC standard (JCB162). This is the default state.
3	JEDEC kernel mode: selects the kernel mode (see Appendix C for kernel mode definition).
4	Fuse display X/-: when a fuse pattern to be edited is displayed, unprogrammed fuses are shown as "X" and programmed fuses are shown as " – ". This is the default state.
5	Fuse display 0/1: when a fuse pattern to be edited is displayed, unprogrammed fuses are shown as "0" and programmed fuses are shown as "1".

NOTE

If you use a value other than the ones specified in the Select Attributes option, when you press the RETURN key after editing a line, the fuse line will be redisplayed with the invalid data. The correct values for representing the programmed and unprogrammed states must be used.

Option	Description
6	End upload with ETX : LogicPak terminates an upload operation (serial data transmission) with an ETX character (ASCII hex 03). This is the default state.
7	End upload with CTRL Z : ends the upload with a CTRL Z .
8	Disable underblow/overblow display: disables this attribute.
9	Enable underblow/overblow display: enables this attribute.

An underblow condition occurs when the programmer RAM indicates that a particular fuse should be blown and the device in the socket shows the fuse to be unblown. An overblow condition occurs when the programmer RAM indicates that a fuse is unblown, yet the part shows it to be blown.

Option	Description
A	Two-pass functional verify: performs the normal two-pass functional verify at V_{CC} voltages above and below nominal.
B	One-pass functional verify: speeds up the testing cycle by doing only a one-pass functional verify at the nominal V_{CC} voltage.

Use the following procedure to access the select attributes from front panel mode.

Procedure	Front Panel Key Sequence	Front Panel Display
1. Access the select attributes function.	SELECT C E START	00
To change any attribute, enter the code number from those given previously.		
2. Change an attribute.	START	0X **
(Where X is the selected option number.)		

Use the following procedure to access the select attributes from the terminal.

Procedure	Terminal Key Sequence	Terminal Display
1. Access the select attributes function.	G	See the following display.

```
Command : G - Select attributes

0 - Echo (full duplex)
1 - No echo (half duplex)

2 - JEDEC full mode (default)
3 - JEDEC kernel mode

4 - Fuse display X/- (default)
5 - Fuse display 0/1

6 - End upload with ETX (default)
7 - End upload with CTRL Z

8 - Disable underblow/overblow display (default)
9 - Enable underblow/overblow display

A - Two pass functional verify (default)
B - One pass functional verify

Options: 0,2,4,6,8,A
```

To change an attribute or attributes from the terminal, SPACE or BACKSPACE (CTRL H) to the appropriate pair of attribute(s) and enter the new value. Terminate an edit session by pressing RETURN if the edited attribute(s) are to be saved or by a CTRL Z if they are not to be saved. If an invalid value is entered, (anything other than the values specified in the Select Attributes option 4 or 5), the line will be repeated, including the invalid data, waiting for the correct value(s) to be entered.

Exit Commands

During terminal mode, a CTRL Z will exit specific operating modes. When using the design adapters, this function is also used to terminate the change, insert, and edit modes.

Use the following procedure to exit an operating mode.

Procedure	Terminal Key Sequence	Terminal Display
1. Exit.	[CTRL] [Z]	Command :

NOTE

The ESC (escape) key is used to terminate terminal control and return control to the front panel. This must be done before removing an adapter or the LogicPak.

Note

If you get a recurring error, call your local Customer Support Center listed at the back of this manual.

Code	Name	Description	Corrective Action
21*	ILLEGAL BIT	Not possible to program the device due to already programmed locations of incorrect polarity.	Erase the device if possible or discard it.
22*	PROGRAM FAIL	The program electronics were unable to program the device.	Either the device is bad or the programming module is inoperative or out of calibration.
25*	NO SOCKET ADAPTER	Either there is no adapter installed or there is a failure in the adapter or programming pak.	Insert appropriate socket adapter or contact your Data I/O Customer Support Center.
30	NO (OR INVALID DEVICE SELECTED)	A device code is entered that either is not on the device list or one that was incorrectly entered.	Enter valid device family and pinout codes (refer to your P/T adapter User Notes.
31*	OVERCURRENT	When any device draws more current than the specified current limit.	A shorted device or hardware error in LogicPak. Substitute a known-good device or consult the troubleshooting section in the maintenance manual.
32*	BACKWARD DEVICE/ V_{CC} OVERCURRENT	The device is in the socket backwards or is a faulty device.	Turn the device around or try a new device.
33	EXT RAM FAIL	PALASM error only. Indicates too much source data for existing RAM.	Expand RAM to 16K or greater or reduce amount of source data.
34	INVALID DEVICE SELECTED	An incorrect family pinout code was entered. This error occurs only in computer remote control.	Enter correct device code.
35	SOURCE EQUATION TRANSLATION ERROR	When the programmer is being controlled by the front panel, the operator is alerted that an error exists in the source equations.	Check for detailed error status by connecting a terminal to the programmer and repeat the translation operation.
36	BEGIN RAM POINTER NOT = 0000	Error usually occurs when changing from one programming pak to another.	Refer to the programmer manual to reset the begin RAM pointer to 0000.

ERROR CODES

Code	Name	Description	Corrective Action
37	INVALID DEVICE-RELATED OPERATION	Verify, program or other illegal operation was attempted with a design adapter installed.	The commands entered are not supported. Choose another entry, or changes to a P/T adapter.
38*	CALIBRATION STEP ERROR	You have either selected an incorrect calibration step or a program operation is attempted prior to exiting calibration.	Exit the calibration mode (refer to your programmer manual.)
63	RAM WRITE ERROR	The programmer is unable to write the intended data in RAM	Failure of the associated RAM chip; replace the failed chip.
65	FIRMWARE SUMCHECK ERROR	The EPROM firmware in the LogicPak or adapter may have changed since the unit was shipped.	Contact Data I/O Customer Support Center. Do not continue operation until the situation is corrected.
70*	DAC ERROR, V_{CC}	Fatal error.	Refer to the troubleshooting section in the maintenance manual or contact your Data I/O Customer Support Center.
71*	DAC ERROR, BIT SWITCH NUMBER 1	Fatal error.	Refer to the troubleshooting section in the maintenance manual or contact your Data I/O Customer Support Center.
72*	DAC ERROR, BIT SWITCH NUMBER 2	Fatal error.	Refer to the troubleshooting section in the maintenance manual or contact your Data I/O Customer Support Center.
73*	DAC ERROR, CE	Fatal error.	Refer to the troubleshooting section in the maintenance manual or contact your Data I/O Customer Support Center.
74*	LOGIC FINGERPRINT TEST VERIFY ERROR	Indicates one of the following Logic Fingerprint errors: 1) Device passed fuse verify but failed Logic Fingerprint test - device defective. 2) Operator has entered wrong test-sum. 3) Device cannot be tested with Logic Fingerprint.	Refer to Appendix B (Test Modes) for Logic Fingerprint Test limitations.

Code	Name	Description	Corrective Action
75	STRUCTURED TEST VERIFY ERROR	The device passed fuse verify but failed structured test — defective device, the vector could be invalid, or the operator may have miskeyed a valid vector.	Check structured test vectors and make sure they are correct or try another device.
76	SELF-TEST ERROR	Indicates failure in the the LogicPak.	Consult the troubleshooting section in the maintenance manual or contact your Data I/O Customer Support Center.
77	SECURITY FUSE PROGRAMMING ERROR	The security fuse cannot be programmed in the installed device.	Try another device.
78*	NO FUSE VERIFY SET	You have tried to program the device with the verify-option mode set for 2.	Select E6 and enter a 0 or 1. Programming will be allowed.
79*	PRELOAD NOT IMPLEMENTED	The preload algorithm is not implemented for this device or the H&L type preload vector not valid.	Check the preload list and verity that the *P* entry in the structured test vector is on the clock pin.
81	PARITY ERROR	The incoming data has incorrect parity.	Check the programmer parity switch and try again.
82	CHECKSUM ERROR	Indicates an incorrect transmission data from a peripheral to the serial port, including fuse data, CRs, STX, etc.	Check all connections of units in the system, data format, and data source, and then try again.
84	INVALID DATA	The programmer received invalid or not enough data characters.	Check the connection of all units in the system, data format and data source, and then try again. Refer to the subsection on Receiving JEDEC Data for more information.
91	FUSE ADDRESS ERROR	Indicates that the fuse number in the L field is higher than the total number of fuses in the part.	Check to be sure that the proper family and pinout code had been entered or that the JEDEC file fuse numbers are correct. Refer to the subsection on Reveiving JEDEC Data for more information.

* These errors do not apply to design adapters.

Functional Testing

Fuse Verify

The fuse verify ensures that the fuse pattern in the device and the programmer RAM are the same, which would indicate that all fuses have been correctly programmed. If the fuse pattern of the device and the programmer RAM correspond, the verify operation will display the sumcheck of the RAM data.

During a fuse verify, the LogicPak compares the fuse pattern of the programmed device, with the fuse pattern in the programmer data RAM. When fuse states in the device do not correspond to those in RAM, the PLDS will display the fuse number, the fuse state in RAM, and whether the first or second pass verify produced the failure. For example, the Model 29 will display:

V1	0001	01
verify pass	fuse number	fuse state in RAM

V1 = first pass

V2 = second pass

The terminal will display:

```
Command : 3 - Verify device

Verifying device
V1 0001 1................
V2 0001 1................

Sumcheck 0002

Command : _
```

Most of the logic paths of a programmable logic device are not tested by the fuse verify. Therefore, a fuse verify will not guarantee that the device will perform its intended function; however, it is a necessary step to ascertain whether or not the device has been programmed correctly.

Structured Vector Test

A structured vector test will be performed on the device automatically after load, programming and during a verify operation only if there are structured test data present in the programmer data RAM. If no data are present, only a fuse verify and a Logic Fingerprint test (if enabled) will be performed.

The structured vector test lets you enter test vectors that stimulate and read device pins, guaranteeing that specific states will be tested. It can also be used to initialize devices before the Logic Fingerprint test. The structured test enables you to uniquely define the inputs and test for desired outputs. The LogicPak applies those inputs and verifies that the desired outputs appear. The structured test cannot be performed unless there are structured test vectors in RAM. With the vectors present, the structured test can be disabled by selecting verify option "1" (see the section on Select Verify Option).

When a device fails a structured test while in the terminal mode, the terminal will display the vector test number and output pin number that failed to show the specified levels. When using the programmer front panel mode, the programmer will display an error code (see Appendix A for a list of error codes). The structured test requires that you write your own test vectors and input them to the programmer RAM using the vector editor, or that you use data development aids such as MMI PALASM, Data I/O ABEL or Data I/O PLDtest to generate the test vectors from specified inputs, outputs, fuse pattern or function tables. A structured test guarantees that certain specified states have been tested.

Test vectors can be generated with the ABEL software program. When designing with ABEL, the test vectors can be input into the source file and simulated in the simulator. Those same vectors can be downloaded to the programmer via a JEDEC file. PLDTest is a software program which is available for users who want to generate test vectors automatically. After the user generates a JEDEC file with ABEL or PALASM, PLDTest will add the generated vectors to a JEDEC file which will test the device more effectively.

Test vectors can also be developed efficiently with the PALASM design adapter. The design aids allow you to enter equations and function tables. Firmware compares the equations and function tables during an operation called "Simulate Function Table." If the equation and function tables are valid, the firmware generates structured test vectors.

The programmer RAM is limited in the number of test vectors it can store (see table). Each vector tests one specific state. For example, if RAM is limited to 50 states and you require testing 100 unique states, you would test 50 states at one time and use multiple loads and test sequences.

RAM Capacity for Structured Test Vectors

Number of Device Pins/RAM	Maximum Number of Vectors
20 pins/4K RAM	50
20 pins/8K RAM	150
20 pins/16K RAM	250
20 pins/64K RAM	250
20 pins/128K RAM	250
24 pins/4K RAM	42
24 pins/8K RAM	128
24 pins/16K RAM	250
24 pins/64K RAM	250
24 pins/128K RAM	250
28 pins/4K RAM	36
28 pins/8K RAM	109
28 pins/16K RAM	250
28 pins/64K RAM	250
28 pins/128K RAM	250

NOTE

Refer to your P/T adapter User Note for any variations to data in this table.

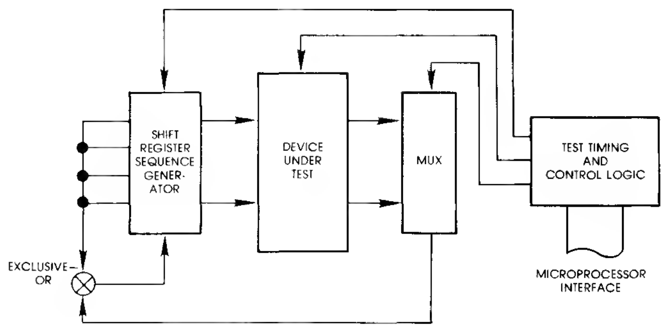
The structured test can be used to: 1) apply all possible input states to a combinational device, 2) force a sequential device through all its state transitions, and 3), because the structured test is performed before the Logic Fingerprint test, present a series of inputs to a device that will drive it to a known initial state so the Logic Fingerprint test can begin from that known state. The latter is required for registered or sequential devices that may power up in an illegal or unknown state.

Structured testing is especially useful with sequential devices because pseudo-random testing algorithms do not achieve the same level of test coverage as they do for combinational devices. Some manufacturers' sequential devices include a preload feature that enables the device to be preset to a known initial state. This preload feature is enabled by specifying special variables in the structured test vector. Sequential devices without this feature will need to be designed and programmed so that structured vectors can force them into a known initial state. This means that you must design the registered (sequential) device so that it can be tested, then write structured test vectors that initialize the device for testing.

Once written, structured vectors can be stored as part of the JEDEC-formatted file (see Appendix C).

Logic Fingerprint Test

Of all testing methods available in logic device programmers, the Data I/O Logic Fingerprint test is the easiest to use and is flexible enough to be applicable to a wide variety of logic devices. To test a device using the Logic Fingerprint test, it is necessary for the LogicPak to first learn a test-sum from a known-good device. Subsequently programmed devices will be tested, and their test-sums will be compared against the reference test-sum. If the test-sums match, the device has passed the Logic Fingerprint test. The following figure shows a block diagram of the Logic Fingerprint test.



The LogicPak learns the reference test-sum from a known-good master device during a load operation. If already known, it can be loaded via the serial port along with programming data (in the JEDEC format) or entered from the programmer keyboard or terminal. If a structured test is being used to initialize devices for the Logic Fingerprint test, the structured test will be performed upon a load operation prior to learning the test-sum. If the device fails the structured test in load, an error code 75 will be displayed (see Appendix A).

The Logic Fingerprint test is enabled by entering the number of test cycles desired. The default number of cycles is zero, which disables the test. Any number of cycles up to 99 can be entered; however, each additional cycle will increase the time of the Logic Fingerprint test. Each cycle takes 1 to 2 seconds, depending on the device type. Only 1 to 8 cycles are necessary in most cases.

The operator can specify the starting vector for the Logic Fingerprint test. The starting vector could be used in two instances: 1) to initialize a sequential device that may not power-on reliably to a known initial state or 2) to test a device that will not respond to the default starting vector of all bits set to zero. The starting vector will enable the Logic Fingerprint test to begin from a known initial state. The starting vector can be up to 28 bits long, depending on the number of pins on the device. A 20-pin device will have a 20-bit starting vector, and a 24-pin device, a 24-bit vector.

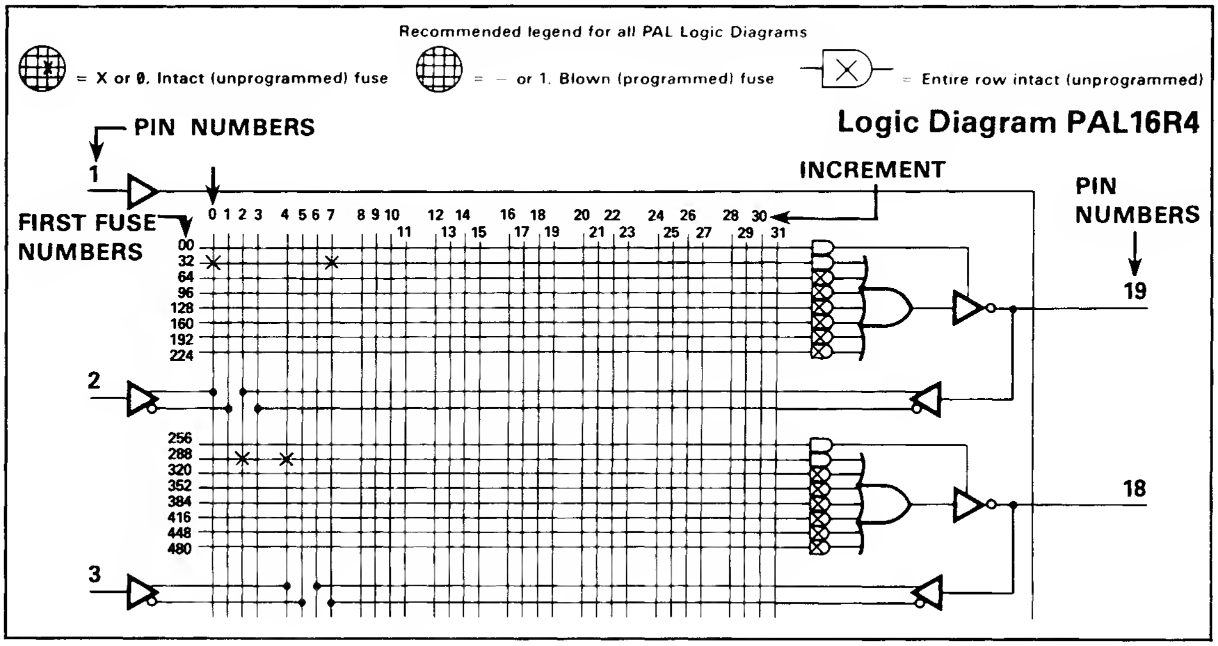
The pseudo-random nature of the input vectors can cause some types of devices with certain fuse patterns to fail the Logic Fingerprint test by giving nonrepetitive test-sums. This does not necessarily indicate a faulty device, but may be an indication that the device is subject to the Logic Fingerprint test limitations. Logic Fingerprint test limitations are described in the following paragraphs. It is very important that you read and understand these limitations.

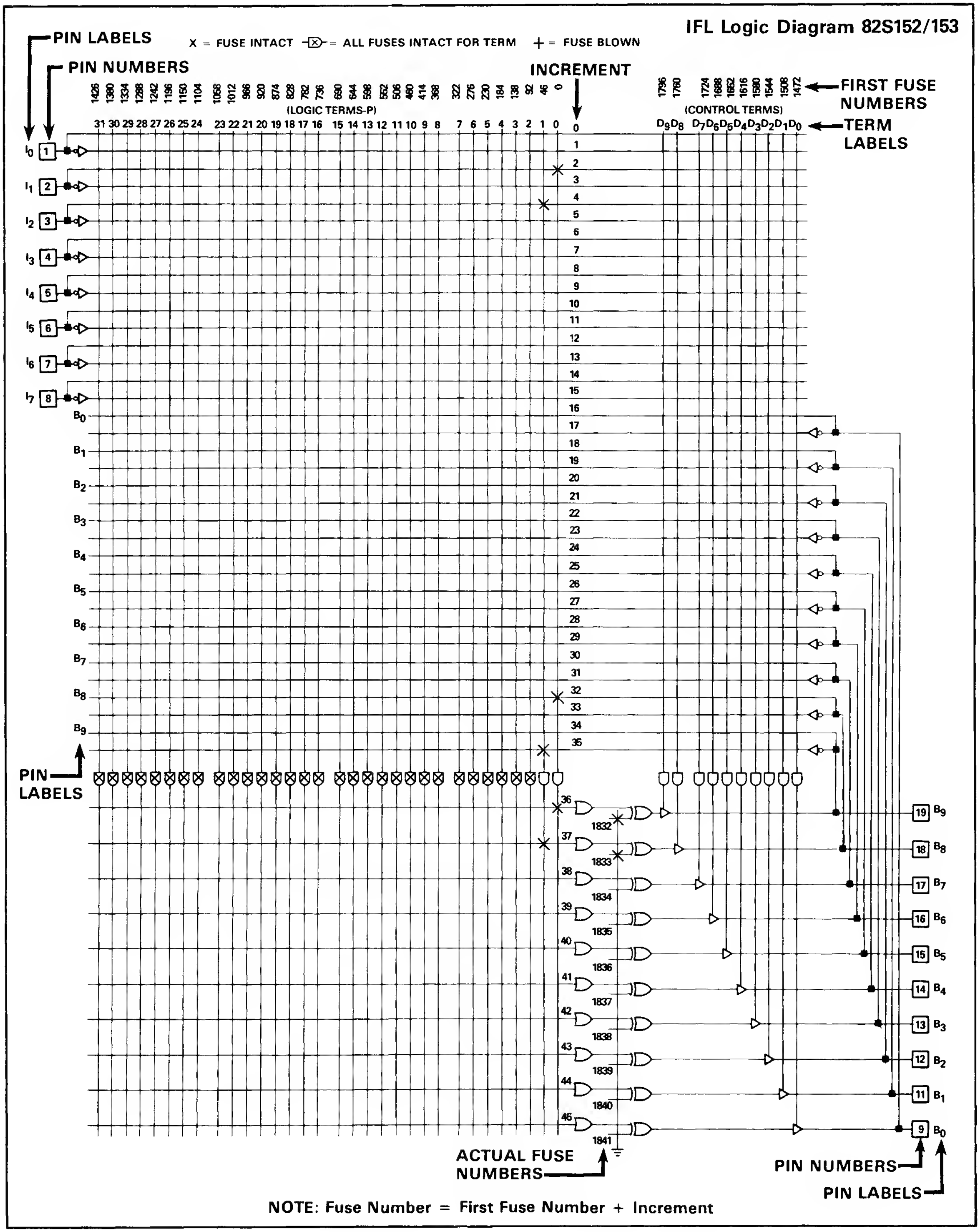
If the Logic Fingerprint test is not suitable for a specific device, it can always be tested by using the structured test.

Logic Fingerprint Test Limitations

Refer to your adapter User Note for device specific Logic Fingerprint test limitations. Following are descriptions of typical PAL and IFL limitations.

Limitation 1: Occurs when devices are programmed so that nonregistered outputs are fed back to product inputs, which results in an oscillation. This condition is shown in the simplified example in the first two figures. The two nonregistered product outputs (pins 19 and 18) in the figure, feed back to the other product's input. If input pins 2 and 3 are both true (i.e., TTL "1"), the device will oscillate. This condition could exist for one product output feeding back to its own input, or for numerous outputs feeding back.




IFL Logic Diagram 82S152/153
PIN LABELS
X = FUSE INTACT
ALL FUSES INTACT FOR TERM
+ = FUSE BLOWN
PIN NUMBERS
INCREMENT
FIRST FUSE NUMBERS
(LOGIC TERMS-P)
(CONTROL TERMS)
TERM LABELS
ACTUAL FUSE NUMBERS
PIN NUMBERS
PIN LABELS
NOTE: Fuse Number = First Fuse Number + Increment

Limitation 2: Occurs when a race condition, caused by incorrect timing pulses in which data goes through a whole string of stages at one step, is programmed into the device. Because the inputs are controlled, it is possible that the race condition will not be critical in the circuit for which the device was designed. Due to the random nature of the inputs during the Logic Fingerprint test, the race condition could appear and cause unstable results. An RS latch is an example of this. The next two figures show the truth table, Boolean equations, fuse map and schematic. Suppose that A, B, and C are all at logic lows, 01 is at a logic high, and 02 is at a logic low. Let B and C go to a logic high simultaneously. The state of S will depend on how fast B and C can propogate through the logic gates. The effect of B will arrive at S first, forcing it low. At a time equal to the propagation delay of the gates later, the effect of C will be seen at S, forcing it back to a logic high. When S was at a logic low, the RS latch changes state and is unaffected when S comes back high. This causes the Logic Fingerprint test to read the wrong values on the outputs, which in turn causes an unstable result.

If the default starting vector of 0 results in a test-sum of FFFF FFFF, select a starting vector other than 0.

RS TRUTH TABLE

R	S	01	02
0	0	1	1
0	1	1	0
1	0	0	1
1	1	a	a

[a]Remains unchanged

BOOLEAN EQUATIONS

$01 = \overline{02 \bullet R}$

$02 = \overline{01 \bullet S}$

$\boxed{\overline{01} = 02 \bullet \overline{A}}$

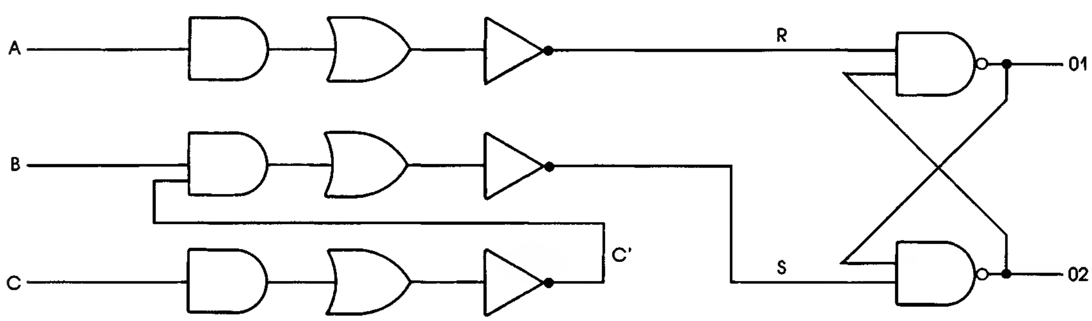
$\overline{02} = 01 \bullet [\overline{B \bullet \overline{C}}]$

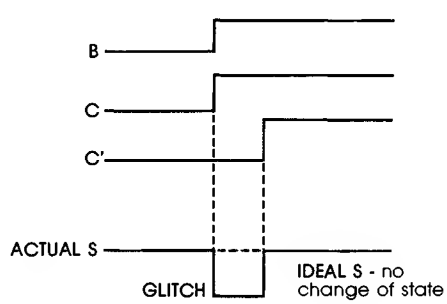
$= 01 \bullet [\overline{B} + C]$

$\boxed{\overline{02} = (01 \bullet \overline{B}) + (01 \bullet C)}$

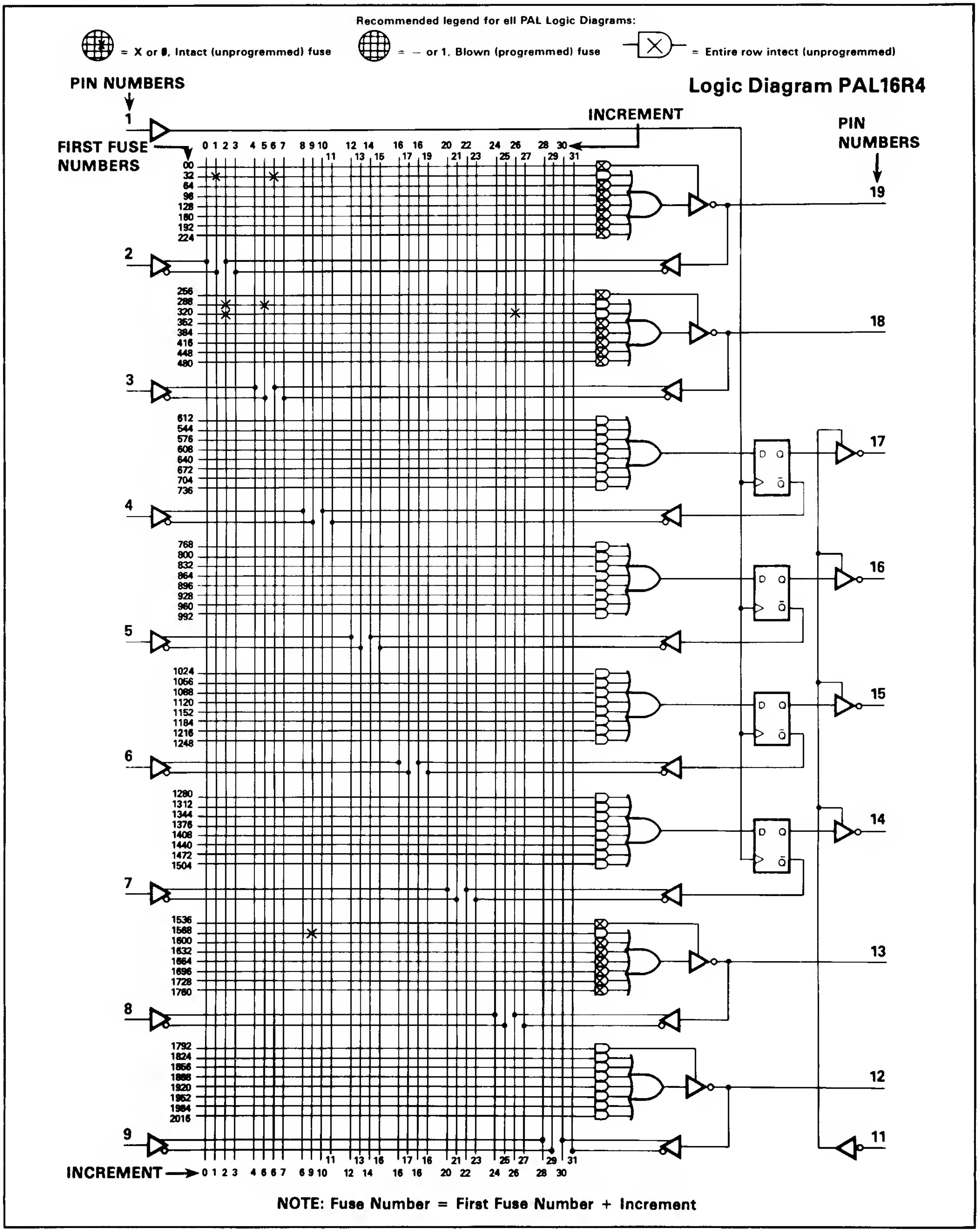
$R = \overline{A}$

$S = \overline{B \bullet \overline{C}}$





Latch changed state when S went low


Recommended legend for all PAL Logic Diagrams:
= X or ●, Intact (unprogrammed) fuse
= — or 1, Blown (programmed) fuse
= Entire row intact (unprogrammed)
Logic Diagram PAL16R4
PIN NUMBERS
FIRST FUSE NUMBERS
INCREMENT
PIN NUMBERS
NOTE: Fuse Number = First Fuse Number + Increment

Limitation 3: Occurs in registered parts only. When using the Logic Fingerprint test, you must start from the same state every time the test is performed. Certain registered devices however, will not power up into the same state every time the test is performed. If the Logic Fingerprint test starts at a different point, it will produce an incorrect result.

To overcome this limitation, the registered outputs must be put into a known state before executing the Logic Fingerprint test. Two methods of doing this are:

1. Dedicate one input line as a preset or reset line for all registered output. A starting vector can then be written to set or clear all registered outputs.
2. If no extra inputs are available to dedicate to a preset/reset line, or a known state (of other than all ones or all zeros) is required, the setup must consist of one or more vectors to force the outputs into the desired state. If more than one vector is needed, the structured test must be used to input the vectors.

NOTE

It is important that you recognize when devices are programmed with these limitations and realize that the Logic Fingerprint test will reject them. These devices can still be tested by using structured test vectors.

JEDEC Format Data Exchange

Fuse data, test vectors, and the Logic Fingerprint test parameters are transmitted between the host computer and the PLDS in the JEDEC format. The JEDEC format is described in detail here. An overview of the format is provided and shown in the next two figures. The first figure shows an example JEDEC transmission and its components.

```
 Command : C - Transmit JEDEC data
PAL12H6            PAL DESIGN SPECIFICATION
P7000               JANE ENGINEER   5-21-83
BASIC GATES
DATA I/O
*
QP20*
QF0384*
L0000
111111101111111111111111
000000000000000000000000
000000000000000000000000
000000000000000000000000
010111111111111111111111
000000000000000000000000
111101111111111111111111
111111110111111111111111
111111111110101111111111
000000000000000000000000*
L0240
111111111111111011011111
111111111111111100111111
111111111111111111111011
111111111111111111111110
111111111111111111101111
000000000000000000000000*
V0001 XXXXXXXXXNXXXXXXXH0N*
V0002 XXXXXXXXXNXXXXXXXL1N*
V0003 00XXXXXXXNXXXXXXLXXN*
V0004 01XXXXXXXNXXXXXXLXXN*
V0005 10XXXXXXXNXXXXXXLXXN*
V0006 11XXXXXXXNXXXXXXHXXN*
V0007 XX00XXXXXNXXXXXLXXXN*
V0008 XX01XXXXXNXXXXXHXXXN*
V0009 XX10XXXXXNXXXXXHXXXN*
V0010 XX11XXXXXNXXXXXHXXXN*
V0011 XXXXXXXX0N00HXXXXXXN*
V0012 XXXXXXXX0N01HXXXXXXN*
V0013 XXXXXXXX0N10HXXXXXXN*
V0014 XXXXXXXX1N00HXXXXXXN*
V0015 XXXXXXXX1N11LXXXXXXN*
V0016 XXXX00XXXNXXXXHXXXXN*
V0017 XXXX01XXXNXXXXLXXXXN*
V0018 XXXX10XXXNXXXXLXXXXN*
V0019 XXXX11XXXNXXXXLXXXXN*
V0020 XXXXXX00XNXXXLXXXXXN*
V0021 XXXXXX01XNXXXHXXXXXN*
V0022 XXXXXX10XNXXXHXXXXXN*
V0023 XXXXXX11XNXXXLXXXXXN*
T01*
S00000000000000000000*
RAB472BBB*
C1BB9*
2642
```

Design Specification Header (PAL12H6 through DATA I/O)

QP20* — Number of Pins

QF0384* — Number of Fuses

L0000 — Fuse Address

Fuse States: 0 = intact, 1 = open

Functional Test Vectors (V0001 through V0023)

T01* — Test Cycles

S00000000000000000000* — Starting Vector

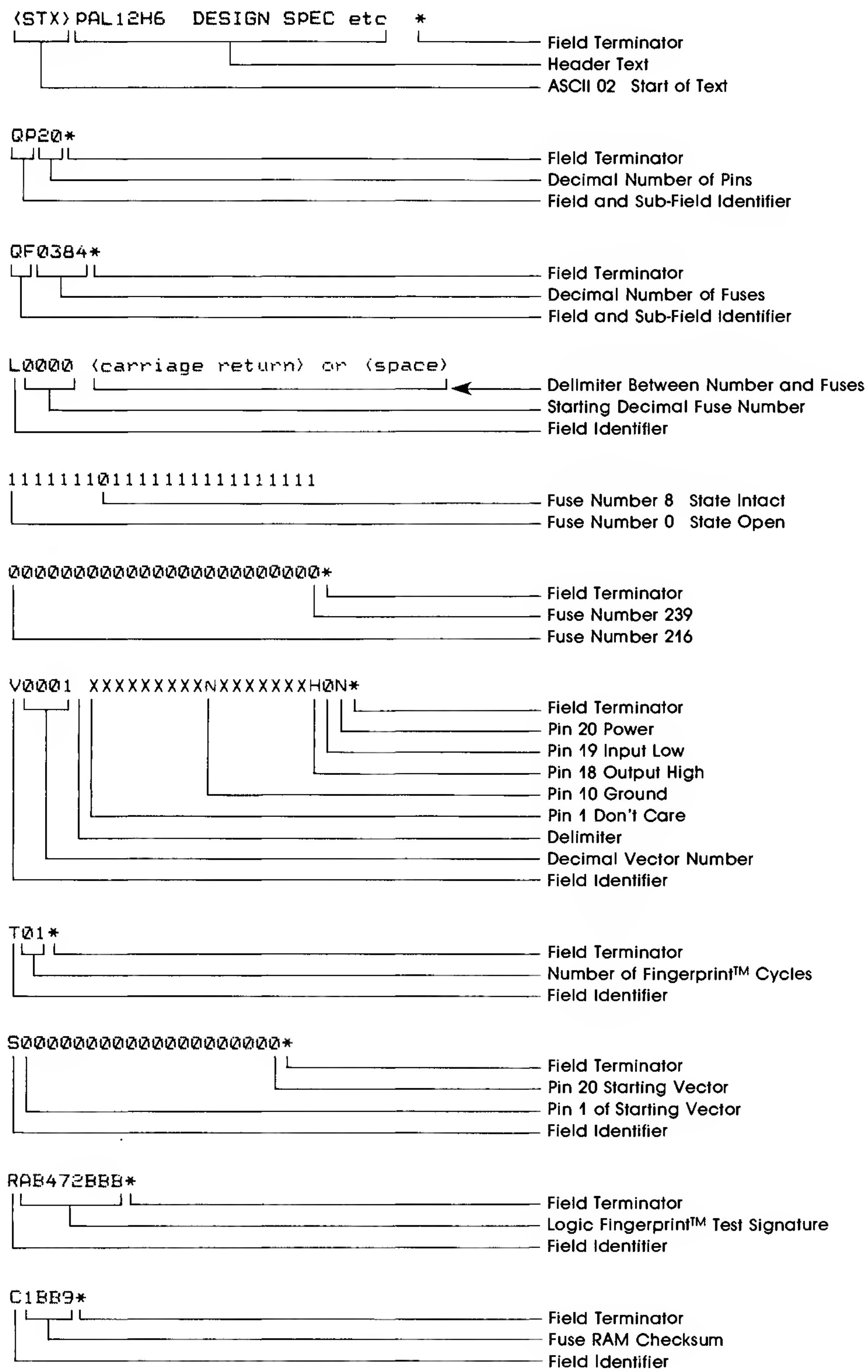
RAB472BBB* — Logic Fingerprint™ Test Signature

C1BB9* — Fuse RAM Checksum

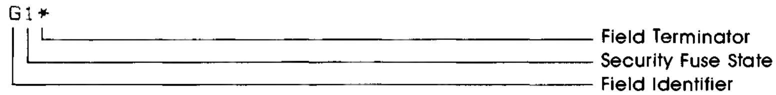
2642 — Transmission Checksum

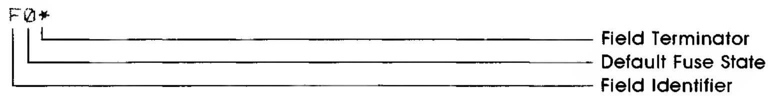
JEDEC FORMAT

The transmission consists of a start-of-text STX character, the various fields, an end-of-text ETX character, and a transmission checksum, as shown in the next figure.



OTHER FIELDS





The transmission checksum is the 16-bit sum of all ASCII characters transmitted between and including the STX and ETX . The parity bit is excluded in the calculation (see the following figure). The transmission checksum computed by the PLDS may be found by examining data RAM addresses 405 and 406, using the programmer's RAM EDIT mode (discussed later in this subsection). Some computer operating systems do not allow a user to control what characters are sent, especially at the end of a line. The transmission checksum may be disabled in this case by sending a dummy checksum of 0 0 0 0 .

```
<STX>*<ETX>002F          02 + 2A + 03 = 2F                            Checksum

random text <return><line feed>                                       = 0000
<STX>TEST*<return><line feed>        02+54+45+53+54+2A+0D+0A = 0183
QF0384*<return><line feed>        51+46+30+33+38+34+2A+0D+0A = 01A7
F0*  <return><line feed>                46+30+2A+20+20+0D+0A = 00F7
L10 101*<return><line feed>   4C+31+30+20+31+30+31+2A+0D+0A = 01A0
<ETX>05C4 <return> other random text                       03 = 0003
                                                                ----
                                                                05C4
```

In general, each field in the format starts with an identifier, is followed by the information, and is terminated with an asterisk. For example, "T 01*" sets the number of Logic Fingerprint test cycles to 1. The design specification header does not have an identifier and must be the first field in the transmission, immediately following the STX

Fuse information is specified by the "QF", "F", "L", and "C" fields. The "QF", "F", and "C" fields are optional.

The "QF" field sets the maximum allowable number of fuses. The "F" field sets the default fuse value. An "F 0*" fills the fuse RAM with 0s, and an "F1*" fills the fuse RAM with 1s. This operation takes a significant amount of time and can lead to an input buffer overflow at high baud rates if handshaking is not used.

The "L" field starts with a decimal fuse number and is followed by a stream of fuse states (1 or 0). The fuse number may include leading zeroes (i.e., "L12" and "L 0012" are the same). A "space" and/or a "carriage return" must separate the fuse number from the fuse states. The stream of fuse states can be as long as desired (up to the maximum allowable fuse number). The fuse data for an entire device, for example, could be sent in one "L" field starting at zero and continuing for all fuses in the device. Spaces and carriage returns may be inserted to make the stream more readable. Each "L" field must be terminated with an asterisk. The "C" field is the sumcheck of the entire fuse RAM (fuse number 0 to maximum fuse number for the selected device), not just the fuse states sent. See the next figure. (The JEDEC term "Fuse Checksum" is the same as Data I/O's term "sumcheck.")

Translator Input Errors

Error	Description	Possible Fields
82	SUMCHK ERR	Transmission checksum
84	INVALID DATA	ETX F L S V
91	I/O FORM ERR	C G L P R T V

The structured test vector information is specified by the "QP", "P", and "V" fields. The "QP" field defines the number of pins on the device. The "V" field starts with a vector number, is followed by a space, then by a series of test conditions for each pin, then is terminated with an asterisk. The test conditions are normally sent in pin number order; however, the "P" field can specify a different sequence. The PLDS JEDEC translator does not validate the test conditions in the vectors (see following table).

Vector Symbol	Definition
0	Drive input low
1	Drive input high
2-9	Drive input to supervoltage # 2-9
C	Drive input low, high, low (clock)
K	Drive input high, low, high (clock)
N	Power pins and outputs not tested
L	Test output low
H	Test output high
Z	Test input or output for high impedance
F	Float input or output
X	Ignore input or output (not defined in JEDEC format)
P	Preload (applied to clock pin)

NOTE

X is not defined in the JEDEC format. The X is treated as an N for outputs and leaves an input at its previously defined state.

The supervoltage test conditions (2 through 9) are used to apply non-TTL levels to certain pins to access special test features. A device could be damaged by improper use of supervoltages.

The Logic Fingerprint test information is specified by the "T", "S", and "R" fields. The "T" field defines the number of test cycles to be performed. The legal values are 0 to 99 decimal. The "S" field defines the starting vector with a series of 1s and 0s for each pin. The "R" field defines the 8-digit hexadecimal Logic Fingerprint test signature.

The "G" field defines the security fuse state. 1 = blown
0 = intact

The "D" field is not sent by new versions of the PLDS JEDEC translator. It has been replaced by the "QF" and "QP" fields and the manual setting of family and pinout codes.

The following information defines a format for the transfer of information between a data preparation system and a logic device programmer. This format provides for, but is not limited to, the transfer of fuse, test, identification, and comment information in an ASCII representation. This format defines the "intermediate code" between device programmers and data preparation systems.

NOTE
This is Data I/O's implementation of the JEDEC (Joint Electron Device Engineering Council) standard (JC-42.1-81-62).

BNF Rules and Standard Definitions

The Backus-Naur Form (BNF) is used in this document to define the syntax of the JEDEC format. BNF is a shorthand notation that follows these rules:

- "::=" denotes "is defined as".
- Characters enclosed by single quotes are literals (required).
- Angle brackets enclose identifiers.
- Square brackets enclose optional items.
- Braces (rounded brackets) enclose a repeated item. The item may appear zero or more times.
- Vertical bars indicate a choice between items.
- Repeat counts are given by a *:n* suffix. For example, a six digit number would be defined as "<number> ::= <digit>:6."

For example, in words, the definition of a person's name reads:

The full name consists of an optional title followed by a first name, a middle name, and a last name. The person may not have a middle name or may have several middle names. The titles consist of: Mr., Mrs., Ms., Miss, and Dr.

The BNF definition is:

```
<full name> ::= [<title>] <f. name>
                {<m. name> } <l. name>
<title> ::= 'Mr.' | 'Mrs.' | 'Ms.' | 'Miss' | 'Dr.'
```

The following standard definitions are used throughout the rest of this document:

```
<digit> ::=  '0'  | '1'  | '2'  | '3'  | '4'  | '5'  | '6'  | '7'  | '8'  |  '9'

<hex-digit> ::= <digit>
              | 'A'  | 'B'  | 'C'  | 'D'  | 'E'  | 'F'

<binary-> ::= '0'  | '1'

<number> ::= <digit> {<digit> }

<del> ::= <space> | <carriage return>

<delimiter> ::= <del> {<del> }

<printable character> ::= <ASCII 20 hex ... 7E hex>

<control character>   ::= <ASCII 00 hex ... 1F hex>
                           | <ASCII 7F hex>

  <STX>               ::= <ASCII 02 hex>
  <ETX>               ::= <ASCII 03 hex>
  <carriage return>   ::= <ASCII 0D hex>
  <line feed>         ::= <ASCII 0A hex>
  <space>             ::= <ASCII 20 hex> | ' '

<valid character> ::=  <printable character>
                     | <carriage return>
                     | <line feed>

<field character> ::= <ASCII 20 hex ... 2A hex>
                     | <ASCII 2C hex ... 7E hex>
                     | <carriage return>
                     | <line feed>
```

Transmission Protocol

Syntax of the Transmission Protocol

<format> ::= <STX> <design spec> {<field>}
<ETX> <xmit checksum>

The transmission consists of a start-of-text (STX) character, various fields, an end-of-text (ETX) character, and a transmission check-sum. The character set consists of the printable ASCII characters and four control characters (STX, ETX, CR, LF). Other control characters should not be used because they can produce undesirable side-effects in the receiving equipment.

The following figure shows a complete JEDEC format file for transmission. The individual components of the format are discussed in the following sections.

```
<STX>
Acme Logic Design  Joan Engineer   Feb. 29 1983
Widget Decode  756-AB-3456 Rev C  Device Mullard 12AX7*
QP20* QF384* G1*
L0000 1111111011 1111111111  1111000000 0000000000
      0000000000 0000000000  0000000000 0000000000
      0000000000 0000000101  1111111111 1111111111
      0000000000 0000000000  0000111101 1111111111
      1111111111 1111110111  1111111111 1111111111*

L0200 1110101111 1111110000  0000000000 0000000000
      1111111111 1111011011  1111111111 1111111110
      0111111111 1111111111  1111111110 1111111111
      1111111111 1111101111  1111111111 1111101111
      0000000000 0000000000  0000*      C1BB9*

V0001 XXXXXXXXXNXXXXXXXHON*   V0002 XXXXXXXXXNXXXXXXXL1N*
V0003 00XXXXXXXNXXXXXXLXXN*   V0004 01XXXXXXXNXXXXXXLXXN*
V0005 10XXXXXXXNXXXXXXLXXN*   V0006 11XXXXXXXNXXXXXXHXXN*
V0007 XX00XXXXXNXXXXXLXXXN*   V0008 XX01XXXXXNXXXXXHXXXN*
V0009 XX10XXXXXNXXXXXHXXXN*   V0010 XX11XXXXXNXXXXXHXXXN*

T02* S10101010101010101010*  R1ACB678F*
<ETX>32EB
```

Design Specification

Syntax of the Design Specification

<design spec> ::= {<field character>}'*'

The design specification is the first field in the format, must be included, and does not have an identifier signalling its start. An asterisk terminates the field. The design specification should consist of:

1. User's name and company
2. Date, part number, and revision
3. Manufacturer's device number
4. Other information

Transmission Checksum

Syntax of the Transmission Checksum

<xmit check-sum> ::= <hex-digit>:4

The transmission check-sum is the 16-bit sum (i.e., modulo 65,535) of all ASCII characters transmitted between and including the STX and ETX. The parity bit is excluded in the calculation.

Fields

Syntax of Fields

```
<field> ::= [<delimiter>] <field identifier>
            {<field character>} '*'

<field identifier> ::=   'C' | 'D' | 'F' | 'G' | 'L' | 'M' | 'N' | 'P' | 'Q'
                       | 'R' | 'S' | 'T' | 'V'

<reserved identifier> ::   'A' | 'B' | 'E' | 'H' | 'I' | 'J' | 'K' | 'O'
                         | 'U' | 'W' | 'X' | 'Y' | 'Z'
```

Each field begins with a single character identifier that identifies the field type. Multiple character identifiers can be used to create sub-fields (i.e., "A1", "A$", or "AB3"). The field is terminated with an asterisk. Therefore, asterisks cannot be imbedded within the field. While not required, carriage returns and line feeds should be used to improve the readability of the format. Reserved identifiers currently have no function and are reserved for future use. The meanings of the field identifiers is given in the following table.

Field Identifiers

A - *	J - *	S - Starting vector
B - *	K - *	T - Test Cycles
C - Checksum	L - Fuse list	U - *
D - Device type	M - Option	V - Test vector
E - *	N - Note	W - *
F - Default fuse state	O - *	X - *
G - Security fuse	P - Pin sequence	Y - *
H - *	Q - Value	Z - *
I - *	R - Resulting vector	

(*indicates reserved for future use)

Device Field (D)

Syntax of the device field:

```
<device> ::= 'D' <field characters> '*'
```

The device field defines the programmable logic device to be programmed.

Example:

D9501*

Fuse Information Fields (F, L, C)

Syntax of the fuse information fields:

<fuse information> ::= [<default state>] <fuse list>
{<fuse list> }
[<fuse checksum>]

<fuse list> ::= 'L' <number> <delimiter>
{<binary-digit> [<delimiter>]} '*'

<default state> ::= 'F' <binary-digit> '*'

<fuse checksum> ::= 'C' <hex-digit>:4 '*'

Each fuse of a device is assigned a decimal number and has two possible states: a zero, specifying a low resistance link, or a one, specifying a high resistance link. Fuse information describing the state of each fuse in the device is given by three fields: the fuse list (L field), the default state (F field), and the fuse checksum (C field).

Fuse states are explicitly defined by the L field. The character *L* begins the L field and is followed by the number of the first fuse for which this field defines a state. The first fuse number is followed by a list of binary values (0 or 1) that indicate the fuse states. When more than one binary value is specified, the additional values are assigned to fuses numbered consecutively from the first fuse number. The L field can be any length desired, and any number of L fields can be specified. If the state for a fuse is specified more than once, the last state replaces all previous ones specified for that fuse.

The F field defines the states of fuses that are not explicitly defined in the L fields. If no F field is specified, all fuse states must be defined by L fields.

The fuse information checksum field is used to detect transmitting and receiving errors. The field contains a number computed by adding 8-bit words containing the fuse states for a device. The 8-bit words are formed as shown in the following figure. Unused bits in the final 8-bit word are set to zero before the checksum is calculated.

word 00	msb							lsb
Fuse No.	7	6	5	4	3	2	1	0
word 01	msb							lsb
Fuse No.	15	14	13	12	11	10	9	8
word 62	msb							lsb
Fuse No.	-	-	-	-	499	498	497	496

Following is an example of full specification of the L, C, and F fields:

F0*L0 01010101* L0008 01010111*
L1000 0101*CF3BA*

Another example, where F and C are not specified:

L0200 0110101010101010 1011
01011101011010001001001 0010*

Structured Functional Test Information (V, P fields)

Syntax of Functional Test Information:

```
<function test> ::= [<pin list>] <test vector>
                    {<test vector> }

<pin list> ::= 'P' <pin number>:N '*'

<pin number> ::= <delimiter> <number>

N ::= number of pins on device

<test vector> ::= 'V' <number> <delimiter>
                  <test condition>:N '*'

<test condition> ::= <digit>     'C'  | 'F'  | 'H'  | 'K'  | 'L'  | 'N'  | 'P'
                               | 'X'  | 'Z'

     <reserved condition> ::=    'A'  | 'B'  | 'D'  | 'E'  | 'G'  | 'I'  | 'J'
                               | 'M'  | 'O'  | 'Q'  | 'R'  | 'S'  | 'T'  | 'U'
                               | 'V'  | 'W'  | 'Y'  | 'Z'
```

Functional test information is specified by test vectors containing test conditions for each device pin. Each test vector contains *n* test conditions where *n* is the number of pins on the device. The following table lists the conditions that can be specified for device pins.

Test Conditions

0 - Drive input low	K - Drive input high, low, high
1 - Drive input high	L - Test output low
2-9 - Drive input to supervoltage # 2-9	N - Power pins and outputs not tested
C - Drive input low, high, low	P - Preload registers
F - Float input or output	X - Output not tested, input undefined
H - Test output high	Z - Test input or output for high impedence

The C and K driving signals are presented after the other inputs are stable. The L, H, and Z tests are performed after all inputs have stabilized, including C and K.

Test vectors are numbered by following the *V* character with a number. The vectors are applied in numerical order to the device being tested. If the same numbered vector is specified more than one time, the data in the last vector replaces any data contained in previous vectors with that number.

The conditions contained in test vectors are applied to the device pins in numerical order from left to right unless specified otherwise with the P field. (The leftmost condition is applied to pin 1, and the rightmost condition is applied to pin 20 of a 20 pin device, for example.) The P field indicates an alternate correspondence between the test conditions and the pin numbers.

The following example uses both the V and P fields to specify functional tests information for a device:

```
P 11 12 13 14 15 16 17 18 19 20
  10  9  8  7  6  5  4  3  2  1*

V0001 C01010101NHLLLHHLHLN*
V0002 C01011111NHLLHLLLHLN*
V0003 C10010111NZZZZZZZZZN*

V0004 C01010100NFLHHLFFLLN*
```

Optional Information Fields (G, S, R, M, N, Q, T)

<option field> ::= <option ident.>
{<field character> } '*'

<option ident.> ::= 'G' | 'S' | 'R' | 'M' | 'N' | 'Q' | 'T'

Optional information can be defined using the G, S, R, M, N, Q, and T fields. Each field must begin with the appropriate character and end with an asterisk; no other restrictions exist. Three examples of optional information fields are given here:

Q*
MFG Acme Semiconductor*
M:1234*

Data I/O uses six optional fields: three for the Logic Fingerprint™ test, a security fuse field, a value field, and a note field.

Logic Fingerprint™ (S, R, T)

Syntax for S, R, T

<starting vector> ::= 'S' <test condition>:N '*'
<resulting vector> ::= 'R' <hex-digit>:8 '*'
<test cycles> ::= 'T' <number> '*'

N ::= number of pins on device

Logic Fingerprint tests are specified by the S, R, and T fields. The S field defines the starting vector for the Logic Fingerprint test. The possible states are 0 (TTL low) and 1 (TTL high). The R field contains the resulting vector or test-sum. The T field denotes the number of test cycles to be run.

Example:

S01000100001110001110110*
R5BCD34A7*
T01*

Security Fuse (G)

Syntax for the Security Fuse Field

<security fuse> ::= 'G' <binary-digit> '*'

The security fuses of certain logic devices may be enabled for programming by sending a 1 in the G field.

Example:

G1*

Values (QF, QP)

Syntax for QF, QP:

<number of pins> ::= 'QP' <number> '*'
<fuse limit> ::= 'QF' <number> '*'

The Q field expresses values or limits required by the receiving device. Two subfields are defined: the P subfield for number of pins on the device, and the F subfield for the number of fuses. These two values enable the receiving device and process the other fields without knowing the device manufacturer or family/pinout code.

Example:

QP24* QF1024 *

Note Field (N)

Syntax of the Note Field:

<note> ::= 'N' <field characters> '*'

The note field is used to place notes and comments in the transfer file.

Example:

N Test Preload*

Other Rules

This section discusses other rules and restrictions concerning the JEDEC standard. Variations on the standard are also discussed.

Transportability

All receiving machines should have a "kernel" mode to ignore all optional fields so that the actual programming data will be transportable. For example, optional fields can be sent to specify additional checksums. A receiving machine in the "kernel" mode could ignore this information, yet still receive the link information required to program the device.

If the optional F field is used to avoid transmitting fuse state data, transportability could be lost. Therefore, whenever practical, data should be transmitted for all links of the device (by using the L field).

Some computer operating systems add control characters after each line making it very difficult to compute the transmission checksum. To disable the transmission checksum, receiving equipment should accept "0000" as a valid checksum.

Syntax for Minimum Transmission to Enhance Portability

<kernel> ::= <STX> <design spec>
<min. fuse information> <ETX>
<xmit check-sum>

<design spec> ::= {<field character>} '*'
<min. fuse information> ::= <fuse list> {<fuse list>}

An example of a "kernel" JEDEC transmission is shown in the following figure.

```
<STX>
Acme Logic Design  Jane Engineer   Feb. 29 1983
Widget Decode  756-AB-3456 Rev C  Device Mullard 12AX7*

L0000 1111111011 1111111111  1111000000 0000000000
      0000000000 0000000000  0000000000 0000000000
      0000000000 0000000101  1111111111 1111111111
      0000000000 0000000000  0000111101 1111111111
      1111111111 1111110111  1111111111 1111111111*

L0200 1110101111 1111110000  0000000000 0000000000
      1111111111 1111011011  1111111111 1111111110
      0111111111 1111111111  1111111110 1111111111
      1111111111 1111101111  1111111111 1111101111
      0000000000 0000000000  0000*

<EXT>0000
```

Variation From Present JEDEC Standard by Data I/O

1. The delimiter after the link number or the vector number may be any combination of carriage returns, line feeds, and/or spaces.
2. The checksum and sumcheck are renamed fuse checksum and xmit checksum respectively.

Variations for 303A LogicPak V01

1. The link number (L field) and the vector number (V field) must both be 4-digit numbers.
2. The delimiter after the link number and the vector number must include a space.
3. Spaces are not allowed between the terminating asterisk of one field and the identifier of the next field.
4. The security fuse ("G" field) requires a space after the binary digit.
5. The kernel mode is not implemented.
6. The D field must have valid family and pinout codes.

   <device> ::= 'D' <hex-digit>:4 '*'

INDEX

U.S. Sales Offices and Representatives
Direct Sales and Service Offices

Data I/O Corporation
10525 Willows Road N.E.
P.O. Box 97046
Redmond, WA 98073-9746
(206) 881-6444
Telex Dom. 15-2167,
Int'l. 4740166 DIO UI

NORTHWEST REGION

Data I/O Corporation
1700 Wyatt Drive
Suite 1
Santa Clara, CA 95054
(408) 727-0641

SOUTHWEST REGION

Data I/O Corporation
2770 South Harbor Blvd.
Suite K
Santa Ana, CA 92704
(714) 662-1182 *(Sales)*
(714) 662-2498 *(Service)*
Telex 755870

EASTERN REGION

Data I/O Corporation
Nashua Road
Route 101A
Amherst, NH 03031
(603) 889-8511 *(Sales)*
(603) 889-8513 *(Service)*
Telex 943431

CENTRAL REGION

Data I/O Corporation
1810 N. Glenville Drive
Suite 108
Richardson, TX 75081
(214) 235-0044
Telex 792474

Representatives

ALASKA, WASHINGTON (TRI-CITIES, VANCOUVER), OREGON

Northwest Test and Measurement
8196 Southwest Hall
Suite 217
Beaverton, OR 97005
(503) 646-9966
1-800-247-5700
Telex 910-467-8775

WESTERN WASHINGTON, SPOKANE

Northwest Test and Measurement
4020 148th Ave. N.E.
Suite F
Redmond, WA 98052
(206) 881-8857

ARKANSAS, LOUISIANA, OKLAHOMA, TEXAS

TESTECH, Inc.
1000 E. Campbell Road
Suite 108
Richardson, TX 75081
(214) 644-5010

4800 West 34th St.
Suite D-6
Houston, TX 77092
(713) 956-0837

1715 Capital of Texas Hwy S.
Suite 107
Austin, TX 78746
(512) 327-7033

OHIO

Electro Sales Associates
1635 Mardon Drive
Dayton, OH 45432
(513) 426-5551

26250 Euclid Avenue
Suite 331
Cleveland, OH 44132
(216) 261-5440

MICHIGAN

Electro Sales Associates
29200 Vassar Road
Suite 505
Livonia, MI 48152
(313) 474-7320

9816 Portage Road
Portage, MI 49002
(616) 323-2416

PENNSYLVANIA, WEST VIRGINIA, EASTERN KENTUCKY

Electro Sales Associates
3740 Mount Royal Blvd.
Allison Park, PA 15101
(412) 487-3801

UPSTATE NEW YORK

DB Associates, Inc.
7000 E. Genesee Street
Building D
Fayetteville, NY 13066
(315) 446-0220

KANSAS, NEBRASKA

Midtec Associates, Inc.
7702 Mize Road
DeSoto, KS 66018
(913) 441-6565
Telex 910-743-4129

MISSOURI

Midtec Associates, Inc.
8460 Lindbergh North
Suite 11
Florissant, MO 63031
(314) 837-5200

FLORIDA

Pen Tech Associates
201 S.E. 15th Terrace
Suite K
Deerfield Beach, FL 33441
(305) 421-4989

1398 Semoran Blvd.
Suite 106
Casselberry, FL 32707
(305) 645-3444

GEORGIA, TENNESSEE

Pen Tech Associates
627 Cherokee Street
Suite 21
Marietta, GA 30060
(404) 424-1931

NORTHERN FLORIDA, ALABAMA, TENNESSEE, MISSISSIPPI

Pen Tech Associates
3322 Memorial Pkwy. S.W.
Suite 36
Huntsville, AL 35801
(205) 881-9298

NORTH CAROLINA, SOUTH CAROLINA

Pen Tech Associates
3709 Alliance Drive
Suite C
Greensboro, NC 27407
(919) 852-6000

MINNESOTA, NORTH DAKOTA, SOUTH DAKOTA

Torkelson Associates
4940 Viking Drive
Minneapolis, MN 55435
(612) 835-2414
Telex 910-576-2740

NORTHERN ILLINOIS, IOWA, INDIANA

Torkelson Associates
108 Wilmot Road, Suite 110
Deerfield, IL 60015
(312) 945-8700
Telex 910-992-1438

INDIANA, WESTERN KENTUCKY

Torkelson Associates
2346 S. Lynhurst
Room B101
Indianapolis, IN 46241
(317) 244-7867
Telex 810-341-3141

WISCONSIN

Torkelson Associates
2840 N. Brookfield Road
Brookfield, WI 53005
(414) 784-7736

ARIZONA, NEVADA

Zeus Electronics, Inc.
1428 E. Pierson Street
Phoenix, AZ 85014
(602) 263-6022
1-800-528-4512
Telex 910-951-1362

NEW MEXICO, EL PASO CO., TEXAS

Zeus Electronics, Inc.
8100 Mountain Road N.E.
Suite 111
Albuquerque, NM 87110
(505) 842-6633
1-800-528-4512

COLORADO, WYOMING

Zeus Electronics, Inc.
3333 Quebec Street
Suite 2950
Denver, CO 80207
(303) 321-4246
1-800-528-4512

UTAH, IDAHO, MONTANA

Zeus Electronics, Inc.
1849 W. North Temple
Suite B105
Salt Lake City, UT 84116
(801) 534-0500
(801) 534-0503

VIRGINIA, WASHINGTON D.C.

SCI-REP, Inc.
9512A Lee Highway
Fairfax, VA 22031
(703) 385-0600
Telex 710-833-0361

NEW JERSEY, EASTERN PENNSYLVANIA

SCI-REP, Inc.
304 Cooper Center
Pennsauken, NJ 08109
(609) 662-5222
Telex 710-892-1297 SCI-REP PESN

Addresses on this list subject to change without notice.

0321/784

International Sales Offices and Representatives

Europe

Data I/O Europa
Vondelstraat 50-52
NL-1054 GE Amsterdam
The Netherlands
(20)186855
Telex 16616 DATIO NL

Data I/O Germany GmbH
Bahnhofstrasse 3
D-6453 Seligenstadt
West Germany
(6182)3088/89
Telex 4184962 DATA D

AUSTRIA

Ing. Ernst Steiner
Hummelgasse 14
A-1130 Wien
Austria
(222)827474
Telex 135026 ES A

BELGIUM

Simac Electronics
Rue du Progres 52
BOITE 3
B-1000 Brussels
Belgium
(2)2192451-3
Telex 23662 SIMEIP B

DENMARK

ITT Multikomponent A/S
Naverland 29
DK-2600 Glostrup
Denmark
(2)451822 or (2)456645
Telex 33355 ITT DK

FINLAND

Instrumentarium Elektroniikka
PL 64 (Vitikka 1)
SF-02630 ESPOO 63
Finland
(358)0905281
Telex 124426 HAVUL SF

FRANCE

M.B. Electronique
606, rue Fourny
Zac de Buc
P.O. Box 31
F-78530, Buc, France
(3)9568131
Telex MB 695414 F

GERMANY

Instrumatic Electronics GmbH
Am Kirchenholzel 14
D-8032 Grafelfing (near Munich)
West Germany
(89)852063
Telex 524298 INSTR D

GREECE

Eltronics Ltd.
2, Alopekis Str. 52
GR-106 75 Athens 139
(1) 7249511/-15
Telex 216589 DARX GR

ISRAEL

TELSYS, Ltd.
12, Kahilat Venetsie Street
Tel-Aviv 69010
Israel
(3)494891-5/(3)494881-2
Telex 32392 TSEE IL

ITALY

Sistrel SPA
Via Giuseppe Armellini 39
I-00143 Roma
Italy
(6)5915551
Telex 680356 SISTREL I

Sistrel SPA
Via P. da Volpedo 59
I-20092 Cinisello Balsamo (Mi)
Italy
(2)6120129 or 6181893
Telex 334643 SISTREL I

NETHERLANDS

Simac Electronics
Veenstraat 20
NL-5503HR Veldhoven
The Netherlands
(40)533725
Telex 51037 SIMAC NL

NORWAY

Teleinstrument A/S
P.O. Box 134
N-1371 Asker
Norway
(2)789460
Telex 72919 TELIN N

PORTUGAL

Decada
Equipamentos de Electronica
e Cientificos, Sarl
Av. Bombeiros Voluntarios, Lote 102b
Miraflores/Alges
1495 Lisboa
Portugal
2103420
Telex 15515 ESPC P

SPAIN

Instrumatic Espanola
Alameda Principal 26
Apartado 151
Malaga 5
Spain
(52)213199/213898
Telex 77131 HAFN E

Instrumatic
Juan Hurtado de Mendoza 9
Madrid-16
Spain
(1)2502577/2507278
Telex 46277 ITIC E

SWEDEN

Macrotek AB
Vallingbyvagen 212, Box 43
S-162 11 Vallingby, Sweden
(8)870190
Telex 12543 MATEK S

SWITZERLAND

Instrumatic SA
5-7, rue du Clos
CH-1207 Geneve
Switzerland
(22)360830
Telex 28667 INSR CH

Instrumatic AG
Weingartenstrasse 9
CH-8803 Ruschlikon
Switzerland
(1)7241410
Telex 56605 INST CH

TURKEY

Data I/O Europa
Vondelstraat 50-52
NL-1054 GE Amsterdam
The Netherlands
(20)186855
Telex 16616 DATIO NL

UNITED KINGDOM

Microsystem Services
P.O. Box 37
Lincoln Road
Cressex Industrial Estate
High Wycombe
Bucks, HP12 3XJ, England
(494)41661
Telex 837187 MICSYS G

International

Data I/O International
10525 Willows Road N.E.
P.O. Box 97046
Redmond, WA 98073-9746
U.S.A.
(206) 881-6444
Telex Dom. 15-2167, Int'l. 4740166 DIO UI

AUSTRALIA

ADELAIDE, AUSTRALIA

Warburton Franki (Adelaide) Pty. Ltd.
322 Grange Road
Kidman Park
South Australia 5025 Australia
3567333
Telex Warfran AA82579

BRISBANE, QUEENSLAND

Warburton Franki (Brisbane) Pty. Ltd.
13 Chester Street
Fortitude Valley
Queensland 4006 Australia
527255
Telex Warfran AA41052

MELBOURNE, VICTORIA

Warburton Franki (Melbourne) Pty. Ltd.
220 Park Street
South Melbourne
Victoria 3205 Australia
699 4999
Telex Warfran AA31370

PERTH, WESTERN AUSTRALIA

Warburton Franki (Perth)
98-102 Belgravie Street
Belmont, 6104
Western Australia
65 7000
Telex Warfran AA92908

SYDNEY, NEW SOUTH WALES

Warburton Franki (Sydney)
1-5 Carter St.
Lidcombe N.S.W.
P.O. Box 394
Australie 2141
648 1711
Telex AA22265

CORPORATE OFFICE

Warburton Franki (Corporate Office)
9 Birnie Ave., P.O. Box 117
Lidcombe, N.S.W. Australia 2141
647-2266
Telex Warfren AA21299

BRAZIL

Cosela
Instrumentos Electronicos Ltda.
Rua de Consolacao, 867
3.° And. - Conj. 32
01301 - Sao Paulo - SP
Brazil
255-1733 or 256-7421 or 231-5548
Telex (011) 30869 CSEL-BR
(011) 38044 CSEL-BR

Cosela
Instrumentos Electronicos Ltda.
Rua Sacadura Cebral, 120
Sala 205 - Saude - 20081
Rio De Janeiro - RJ
(021) 283-2036
Telex (021) 33019

CANADA

Allan Crawford Associates, Ltd.
6503 Northam Drive
Mississauga, Ontario L4V 1J2
(416) 678-1500
Telex 06 968769

881 Lady Ellen Place
Ottawa, Ontario K1Z 5L3
(613) 722-7682
Telex 053 3600

7018 Cote De Liesse
St. Laurent, P.Q. H4T 1E7
(514) 731-8564
Telex 05-824944

1935-30th Avenue N.E.
Calgary, Alberta T2E 6Z5
(403) 230-1341
Telex 03 821186

3795 William Street
Burnaby, B.C. V5C 3H3
(604) 294-1326
Telex 04 54247

192 Joseph Zatzman Drive
Dartmouth, Nova Scotia B3B 1N4
(902) 463-9360
Telex 019 31604

15043A 118th Avenue
Edmonton, Alberta T5V 1H9
(403) 451-4893

CHINA

Dorado Co.
P.O. Box 4155
Seattle, WA 98104
583-0000

International P.O. Box 9051
Beijing, China
Telex: 329473 (Burgess Sea)

HONG KONG

Eurotherm (Far East) Ltd.
49-51 Wong Chuk Hang Rd.
Flat A & B, 19/F
Derrick Industrial Bldg.
Aberdeen, Hong Kong
5-546391
Telex 72449 EFELD HX

INDIA

Transmarketing Private, Ltd.
Sterling Center
16/2 Dr. Annie Besant Rd.
Bombay 400-18
India
022 4921874/022, 4920320
Telex 011-5424 TMTIN

59, Millers Road
Benson Town
Bangalore
India 560-046

JAPAN

Data I/O Japan Company, Ltd.
Ginza Orient Building, 6F,
8-9-13, Ginza, Chuo-ku
Tokyo 104 Japan
(03) 574-0211
Telex 2522685 DATAIO J

S. KOREA

Elcom System, Inc.
Jinduk Bld. Room 302
55-15 Nonhyun-Dong
Kangnam Ku
Seoul, Korea
555-5222 or 557-3836
Telex ADUCEL K25227

MALAYSIA

GEA Technology PTE., Ltd.
Units 1003 to 1008, Block 3, 10th Floor
PSA Multi-Storey Complex
Pasir Panjang Road
Singapore 0511
2729412
Telex RS 37162 Answerback Geasin

MEXICO

Christensen, S.A.
Guillermo Prieto 76-304
Col. San Rafael
Delegacion Cuahutemoc
06470-Mexico, D.F.
546-25-95/546-29-55
Telex 017-75612 Mycome

NEW ZEALAND

Wellington-New Zealand:
Warburton Franki, Ltd.
42-43 Oxford Terrace
Lower Hutt, New Zealand
693-016
Telex Warfran NZ 3824

Warburton Franki, Ltd.
P.O. Box 9301; Newmarket
142 Broadway
Auckland, New Zealand
504-458
Telex Warfran NZ 3824

PHILIPPINES

Data I/O International
10525 Willows Road N.E.
P.O. Box 97046
Redmond, WA 98073-9746
(206) 881-6444
Telex Dom. 15-2167/Int'l. 4740166 DIO UI

SINGAPORE

GEA Technology PTE., Ltd.
Units 1003 to 1008, Block 3, 10th Floor
PSA Multi-Storey Complex
Pasir Panjang Road
Singapore 0511
2729412
Telex RS 37162 Answerback Geasin

SOUTH AFRICA

Electronic Building Elements (PTY) Ltd.
P.O. Box 4609
Pine Square, 18th Street
Hazelwood
Pretoria, South Africa
46-9221/7
Telex 3 0181 SA
Telegrams Elbilem

SOUTH & CENTRAL AMERICA

Data I/O International
10525 Willows Road N.E.
P.O. Box 97046
Redmond, WA 98073-9746
U.S.A.
(206) 881-6444
Telex Dom. 15-2167, Int'l. 4740166 DIO UI

TAIWAN

Surtek International, Inc.
315 Fu Hsing N. Road
104, Taiwan R.O.C.
(2) 713-4022
Telex 23756 MULTIIC

THAILAND

Dynamic Supply Engineering R.O.P.
12 Soi Pasara 1, Sukhumvit 63
Bangkok-11, Thailand
Tel. - 3914434, 39228532
Telex 82455 DYNASUP TH

Addresses on this list subject to change without notice.

0321/784